UDC 004.81

Applicability of Weyuker's Properties on OO Metrics: Some Misunderstandings

Sanjay Misra and Ibrahim Akman

[1]Faculty of Engineering, Department of Computer Engineering
Atilim University Ankara,Turkey
{smisra,akman}@atilim.edu.tr

Abstract. Weyuker's properties have been suggested as a guiding tool in identification of a good and comprehensive complexity measure by several researchers. Weyuker proposed nine properties to evaluate complexity measure for traditional programming. However, they are extensively used for evaluating object-oriented (OO) metrics, although the object-oriented features are entirely different in nature. In this paper, two recently reported OO metrics were evaluated and, based on it; the usefulness and relevance of these properties for evaluation purpose for object-oriented systems is discussed.

Keywords: software complexity measures; weyuker's properties; evaluation criteria; object oriented metrics.

1. Introduction

A newly proposed complexity measure is acceptable only when its usefulness has been proved by validation process. The purpose of validation is also to prove the usefulness of the attribute measured by the proposed metric. Today, existing literature provides varieties of proposals [6], [7], [8], [11] for validation process of complexity metrics. Among these, Weyuker's properties [11] play an important role in evaluating software complexity measures. These properties are used to evaluate the robustness of a measure and in turn lead to the definition of good notions of software complexity. With the help of these properties, one can determine the most suitable measure among different available complexity measures. Weyuker's [8] proposed these properties to evaluate complexity measure when only traditional (i.e. procedural) programming languages were in use. These properties were also used by some researchers for evaluation of popular object oriented metrics (for example Chidamber's metrics [2] and Kapsu's metric [5]) although the object-oriented features are entirely different in nature. Due to the high popularity and acceptance of Chidamber's metrics, Weyuker's properties are assumed to be accepted as an evaluation criterion for OO metrics. Recently, Sharma et al. [7] proposed a complexity metric based on different constituents of the

components like inheritance of classes, methods and attributes and Aggarwal et al. [1] proposed two object oriented software design metrics for exception handling. Both of these studies used Weyuker's properties for the theoretical evaluation. In other words, more and more researchers are adopting these properties for the evaluation of their new object oriented metrics. Although these properties are simple and straightforward it is observed that, sometimes the authors of new measures misunderstand the interpretation of these properties. It is mainly because these properties were not developed for object oriented programming languages at the beginning. Therefore, it is important to discuss the interpretation, relevance and the applicability of Weyuker's properties in OO domain. This is the key reason for our present work.

The organization of the paper is as follow. In the next section, the Weyuker's properties are reviewed in short. Section three considers two cases and analyzes how these properties are misunderstood for some of its axioms for evaluating OO metrics. The last two sections constitute discussions and conclusions.

2. Weyuker's Properties

Weyuker proposed [11] nine properties for procedural languages. For the sake of convenience for readers and developers of the new OO metrics, these properties for OO domain are presented by considering class as a basic unit instead of program bodies.

Property 1. $(\exists P)\ (\exists Q)\ (|P| \neq |Q|)$,.where P and Q are the two different classes.

This property states that a measure should not rank all classes as equally complex.

Property 2: Let c be a non-negative number, and then there are only finite number of classes and programs of complexity c.

There are only a finite number of classes of the same complexity.

Property 3: There are distinct class P and Q such that $|P| = |Q|$

This property states that there are multiple classes of the same complexity.

Property 4: $(\exists P)\ (\exists Q)\ (P \equiv Q \ \&\ |P| \neq |Q|)$.

This property states that implementation is important. If there exist classes P and Q such that they produce the same output given the same input.

Property 5: $(\forall P)\ (\forall Q)\ (|P| \leq |P; Q| \ \&\ |Q| \leq |P; Q|)$.

This property states that if the combined class is constructed from class P and class Q, the value of the class complexity for the combined class is larger than the value of the class complexity for the class P or the class Q.

Property 6a: $(\exists P)(\exists Q)(\exists R)(|P| = |Q|) \ \&\ |P;R| \neq |Q; R|)$.

6b: $(\exists P)\ (\exists Q)\ (\exists R)\ (|P| = |Q|).\ \&\ |R; P| \neq |R; Q|)$.

This property states that if a new class is appended to two classes which have the same class complexity, the class complexities of two new combined classes are different or the interaction between P and R can be different than

interaction between Q and R resulting in different complexity values for P+ R and Q + R.

Property 7: There are program bodies P and Q such that Q is formed by permuting the order of the statements of P, and (| P | ≠ | Q |).

This property states that permutation of elements within the item being measured can change the metric values. The intent is to ensure that metric values change due to permutation of classes.

Property 8: If P is renaming of Q, then | P | = | Q |.

This property requires that when the name of the class or object changes it will not affect the complexity of the class. Even if the member function or member data name in the class change, the class complexity should remain unchanged.

Property 9: (∃P) (∃Q) (| P | + | Q |).< (| P; Q |).

This property states that the class complexity of a new class combined from two classes is greater than the sum of two individual class complexities. In other words, when two classes are combined, the interaction between classes can increase the complexity metric value.

3. Case Study: Weyuker's Properties and Their Applicability on Recently Proposed Measures

This section analyzes the applicability of Weyuker's properties for two object oriented metrics proposed by Sharma et al. [10] and Aggarwal et al. [1]. Both papers are available online and, therefore, the details of these metrics are not given here. Although their contribution for developing such a metric is valuable for the software community, it is clear that, they misunderstood and misinterpreted some of these properties.

Weyuker's first four properties are general in nature and assumed to be satisfied by any sensible measure. In the measure of Sharma et al. [10], the proof for second property is explained as "as a component will have only the finite number of methods and variables, which always will have a finite value of the complexities, thus resulting a finite complexity for the entire component." However, Weyuker's second property states that "if c is a non negative number, then there are only finitely many programs for which the metric values are c." Clearly, there is a misunderstanding in the explanation of Weyuker's second property. As noted by the Weyuker, c is assumed to be the possible largest number and should be represented as an upper bound on the size of program bodies. Therefore, Sharma et al. [10] should show how, for a given value of complexity, there are finitely many number of program bodies for their measure.

Weyuker's fifth property is the property of monotonic. It states that the metric value for the combination of classes/components can never be less than the metric value for either of the components/classes. It is reasonable for any metric that the complexity should increase when more logic in the code or functionality inside the method is added. Sharma et al.'s [10] component

complexity (CC) is satisfied by this property. However, complexity measures proposed by Aggarwal et al. [1], Number of Catch Blocks per Class (NCBC) and Exception Handling Factor (EHF), do not satisfy this property. They applied Weyuker 5th property on their measure as; (pp.130),

u(p)= p and u(q) =q, then u(p+q)=p+q-m,

where u is the metric, p and q are classes/programs, and m is the catch blocks common to a class. They suggested that in some cases

u(p)≥ u(p+q) and u(q)≥u(p+q).

However, it seems that there is some misunderstanding in their proof. For example, consider the case given by the authors on page 124. For class A and B they measured their complexities as NCBC(A) = 2/3 and NCBC(B)=4/6, i.e., u(p)= 2/3 and u(q)= 2/3. The number of catch blocks for A and B are 3 and 6 respectively in their example. Therefore, the maximum value of m (common catch block) can not be greater than 3. Then

u (p+q) =p+q-m= (2/3)+(4/6)-3= -1.6

It is not desirable that a measure gives a negative value after combination of two classes/objects. Therefore, there must be some misunderstanding and misinterpretation for this property. The similar discussion is also valid for EHF.

Weyuker's property six is satisfied by CC, NCBC and EMF.

Weyuker's property seven states that permutations of program statements can change the metric value. For NCBC and EMF, Aggarwal et al. [1] used Chidamber et al.'s [2] approach and didn't consider property seven because Chidamber et al. did not use it. In the original Chidamber's paper, they argued that this property is not useful in calculating the complexity of a class because order of statements within the class is not important in calculating the complexity. However, in our opinion, if the class complexity is calculated by the adding the method complexities since the order of statements are important in this case. Therefore, one should be careful when using this property. Sharma et al. [10] also interpreted the property seven as "permutation on component's constituents does not effect on the metric value. Therefore it satisfies 7th property." It is again another misunderstanding of property seven. This is just opposite to the corresponding original Weyuker's seventh property.

Weyuker's eight property is related to renaming, so is satisfied by CC, NCBC and EHF.

Weyuker's ninth property states that if two classes/components are combined, the interaction between them can increase the complexity metric value. The Authors of NCBC and EHF rejected this property only by saying that "it is not applicable for object oriented metrics, as stated by Chidamber et

al. [2]". On the other hand, such type of indication in the original paper of Chidamber et al. [2] is not found. They only argued that this property may not be an essential feature of OO software design. It appears only a suggestion and they themselves used this property for evaluating for their all measures. Further, the usefulness of this property for OO metrics has been discussed and proved by several researchers [4], [9] and [12]. The CC [10] metric is satisfied by this property. Sharma et al. [10] interpreted this property, for the proof of their measure, as "In the object oriented perspective, modularity reduces the complexity. Thus the total complexities of the two modules will be lesser than the complexity of the combined module, which satisfies the last property." This is again a misunderstanding for this property by the authors. In the case of OO software development it is observed that complexity increases when classes are divided into more classes [2]. It is because when the classes or components are combined into a single class or component, several features which are common in them, after combination, reduces the complexity. Therefore, in general, the object oriented metrics [2] [5] do not satisfy this property.

4. Discussion

It is observed that there are some misunderstandings for the use and interpretation of Weyuker's properties for OO metrics. This observation is demonstrated by using complexity measures proposed by Aggarwal et al. [1] and Sharma et al. [10]. In their article, Sharma et al. proposed that CC satisfies all Weyuker's properties. The reason behind this observation is that some of the authors think that all Weyuker's properties should be satisfied for their metrics. This is not true. It is not necessary to satisfy all the properties since some of the properties are themselves contradictory, like properties five and six according to the rules of measurement theory. In the original work of Weyuker's, she even didn't find a single complexity measure which is satisfied by all properties. Further, a complexity measure satisfied by all properties arises a question mark on the utility of the measure [3].

Some important points/observations regarding the Weyuker's properties, and object oriented metrics (especially at class level) are also suggested. These points are only general observations and are not rule of thumps.

An object oriented metric should satisfy Weyuker's properties 1, 2, 3, 4, 6 and 8. This is because of the following facts,

1. Since not every class can have the same value for a metric. (Weyuker's property 1)
2. Since the universe of discourse deals with at most a finite set of applications, each of which has a finite number of classes. (Weyuker's property 2)
3. Since in every complexity measure, one can find the number of classes, which can have same metric value. (Weyuker's property 3)

4. Since it is possible that even though two-class design performs the same function, the details of the design matters in determining the metric for the class. (Weyuker's property 4)
5. Since, in the interaction between two classes P and R can be different than interaction between Q and R resulting in different complexity values. (Weyuker's property 6)
6. Since, there are no effects in complexity by renaming. (Weyuker's property 8)

The observations say that proof of properties 5, 7 and 9 varies from metric to metric. More specifically, property 5 should be satisfied by object oriented measures, however in some exceptional cases, this property is not satisfied. Lack of cohesion and depth of inheritance tree [2] are the examples of the exceptional cases.

In OOP, a class is an abstraction of the problem space, and the order of statements within the class definition has no impact on eventual execution or use .This is the reason that most of the OO metrics should not be satisfied by property 7. On the contrary, if the class complexity is calculated by adding the method complexity (because in the method complexity orders of statements are important) then this property is satisfied.

Weyuker's Property 9 has received a mixed response regarding its applicability to object oriented software metrics and, on the contrary to past beliefs, the relevance of this property to object oriented systems is brought out [9].

5. Conclusion

Although Weyuker properties attracted criticism, they are gaining popularity as an evaluation criterion for newly proposed OO measures. It is mainly because of the fact that these properties are simple in nature and easy to implement. However, one should be careful when applying these properties in OO domain since these properties were basically proposed for procedural languages. Further, evaluation through Weyuker's properties can only be achieved up to some degree because these properties are known to be describing only necessary but not sufficient conditions for any metric evaluation purpose [3]. Therefore, if any object oriented or component based metric is evaluated by Weyuker's properties then it should also be evaluated by other measurement criteria to fulfill sufficiency.

6. References

1. Aggarwal K.K., Singh,Y. Kaur, A. and Melhotra, R.: Software Design Metrices for object orinted Software, Journal of Object Technology, vol.6.no.1. pp. 121-138, (2006).

2. Chidamber, S.R and Kemerer, C.F.:A Metric Suite for Object Oriented Design,, IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476-493, (1994).
3. Cherniavsky, J.C. and. Smith, C.H.: On Weyuker's Axioms for Software Complexity Measures, IEEE Transactions on Software Engineering, vol. 17, pp. 636-638, (1991).
4. Gursaran and Roy, G.: On the Applicability of Weyuker Property Nine to Object Oriented Structural Inheritance Complexity Metrics, IEEE Trans. Software Eng., vol. 27, no. 4, pp. 361-364, (2001).
5. Kapsu K., Shin, Y.,Chisu W.: Complexity Measures for Object-Oriented Program Based on the Entropy, Software Engineering Conference, 1995. Proceedings, 1995 Asia Pacific 6-9 Dec. pp.127 – 136, (1995).
6. Mouchawrab, S., Briand L.C., Labiche, Y.: A Measurement Framework for Object-Oriented Software Testability, Information and Software Technology, Vol.47, pp. 979-997, (2005).
7. Poels, G., Dedene G.: Distance-based Software Measurement: Necessary and Sufficient Properties for Software Measures, Information and Software Technology Vol. 42, 1, pp. 35-46, (2000).
8. Stockhome, S.G., Todd, A.R., Robinson, G.A.,: A Framework for Software Quality Measurement, IEEE Journal on Selected Areas in Communications,Vol.8,no.2, pp.224-233. (1990).
9. Sharma ,N., Joshi, P. and Joshi, R.K.: Applicability of Weyuker's Property 9 to Object Oriented Metrics, IEEE Transactions on Software Engineering, vol. 32, no. 3, pp. 209-211, (2006).
10. Sharma, A., Kumar, R and Grover, P.S.: Empirical Evaluation and Critical Review of Complexity metrics for Software Components, Proceedings of the 6th WSEAS Int.Con.on SE, Parallel and and disributed systems. pp. 24-29, (2007).
11. Weyuker, E. J.: Evaluating Software Complexity Measure. IEEE Transaction on Software Complexity Measure, 14(9), pp. 1357-1365,(1988).
12. Zhang,L., and Xie, D.: Comments on 'On the Applicability of Weyuker Property Nine to Object Oriented Structural Inheritance Complexity Metrics', IEEE Trans. Software Eng., vol. 28, no. 5, pp. 526-527, (2002).

Sanjay Misra is Assistant Professor in Department of Computer Engineering, Atilim University, Ankara, Turkey. He obtained M.Tech. Degree in software engineering from Motilal Nehru National Institute of Technology, India and P.hd. from University of Allahabad, India. Presently he is working in the area of Software Engineering, especially on software metrics and measurement theory. His areas of interests are Software measurement, software architecture, service oriented architecture, XML technologies, and Web Services. He published more than 30 research papers in reputed National and International Journals and Conference proceedings.

K. Ibrahim Akman is professor and the chairman of Computer Engineering Department at Atilim University. He obtained his Ph.D. in operations research from Lancaster University (U.K.) in 1984. Dr. Akman has served on the editorial boards of Electronic Journal of e-Government and International Journal of Information Technology and Management (IJITM). Software piracy, e-Government, human resource management and data compression

are amongst his current fields of interest. He published over 50 articles in conferences and journals including Studies in Educational Evaluation, Computational Statistics and Data analysis, Microelectronics and Reliability, Journal of Information Science, Government Information Quarterly, International Journal of Information Management and Behaviour and Information Technology.

Received: September 6, 2007; Accepted: February 20, 2008.