Applicability of Weyuker's Property 9 to Inheritance Metric

Sandip Mal
Birla Institute of Technology
Department of CSE
Ranchi-835 215, India

Kumar Rajnish
Birla Institute of Technology
Department of IT
Ranchi-835215, India

ABSTRACT

In the metric suite for Object-Oriented design put forward by Chidamber and Kemerer it is observed that Weyuker property 9 is not satisfied by any of the structural Inheritance Complexity Metrics. The same is also observed for the candidate structural inheritance complexity metric by Brito and Carapuca, Li's inheritance metric suite, Rajnish and Bhattacherjee inheritance metric. This paper presents two new inheritance metrics: one is ICC (Inheritance Complexity of Class) measured at the class level, which does not satisfies Weyuker property 9 and another one is ICT (Inheritance Complexity of Tree) measured at the tree level, which satisfies the Weyuker property 9 (Interaction Increases Complexity). Examples supporting the applicability of the property are also presented.

General Terms

Complexity, Design, Methods.

Keywords

Classes, Inheritance Tree, Metrics, Object-Oriented.

1. INTRODUCTION

In order to provide mathematical rigour and an axiomatic basis to complexity metrics, necessary, though not sufficient, properties have been set forth by Weyuker [1], and others [12] [13]. Several researchers have evaluated their metric suites against Weyuker properties [2] [3] [4] [5] [6] [14] while others are skeptical about these properties [7] [10] [18]. Research has also been conducted regarding inheritance metrics by Rajnish and Bhattacherjee in [9] [15] [16] [17].

Weyuker property 9 has been the most controversial property with respect to Object-Oriented metrics. The property tries to capture the complexity occurring due to interaction during composition. It states that there exist compositions which result in complexity greater than the sum of the complexities of individual components that are composed.

The rest of the paper is organized as follows: Section 2 presents Weyuker's properties and assumptions. Section 3 presents Inheritance Metrics and analytical evaluation results. Section 4 presents new Inheritance Metrics (ICC and ICT) along with supported examples. Finally Section 5 deals with conclusion and future work.

2. WEYUKER PROPERTIES

The basic nine properties proposed by Weyuker [1] are listed in Table 1. The notations used are as follows: P, Q, and R denote combination of classes P and Q, μ denotes the chosen metrics, μ (P) denotes the value of the metric for class P, and P≡Q (P is equivalent to Q) means that two class designs, P and Q, provide the same functionality. The definition of combination of two classes is taken here to be same as suggested by [2], i.e., the combination of two classes results in another class whose properties are union of the properties of the component classes. Also, combination stands for Weyuker's notion of "concatenation".

Table 1: Weyuker Property

No	Name	Description
1	Noncoarseness	$(\exists P)(\exists Q)(\mu(P) \neq \mu(Q))$
2	Granularity	Let c be a non-negative number. Then there are finitely many programs of complexity c
3	Non-Uniqueness	There are distinct programs P and Q such that $\mu(P) = \mu(Q)$
4	Design Details Matter	$(\exists P)(\exists Q)(P \equiv Q)$ and $\mu(P) \neq \mu(Q)$
5	Monotonicity	For all classes P and Q such that $\mu(P) \leq \mu(P+Q)$ and $\mu(Q) \leq \mu(P+Q)$
6	Nonequivalence of interaction	$(\exists P)(\exists Q)(\exists R)$ such that $\mu(P) = \mu(Q)$ does not imply that $\mu(P + R) = \mu(Q + R)$
7	Interaction among statements	Not consider for Object-Oriented metrics.
8	No change on renaming	If P is a remaining of Q then $\mu(P) = \mu(Q)$
9	Interaction increases complexity	$(\exists P)(\exists Q)(\mu(P) + \mu(Q) < \mu(P+Q)$

Analytical evaluation is required so as to mathematically validate the correctness of a measure as an acceptable metric. For example, Properties 1, 2, and 3 namely Non-Coarseness, Granularity, and Non-Uniqueness are general properties to be satisfied by any metric. Property 9 of Weyuker will not normally be satisfied by any metric for which high values are an indicator of bad design measured at the class level. In case it does, this would imply a case of bad composition and the classes, if combined, need to be restructured. Having analytically evaluated a metric, one can proceed to validate it against data.

Assumptions. Some basic assumptions used in section 4.2 have been taken from Chidamber and Kemerer [2] regarding the distribution of methods and instance variables in the discussions for the metric properties.

Assumption 1:

Let X_i= the number of methods in a given class i

Y_i= the number of methods called from a given method i

Z_i= the number of instance variables used by a method i

X_i, Y_i, Z_i are discrete random variables, each characterized by some general distribution function. Further, all the X_i's are independent and identically distributed within the program. The same is true for all Y_i's and Z_i's. This suggests that the number of methods and variables follow a statistical distribution that is not apparent to an observer of the system. Further, that observer cannot predict the variables and methods of one class based on the knowledge of the variables and methods of another class in the system.

Assumption 2: In general, two classes can have a finite number of "identical" methods in the sense that a combination of the two classes into one class would result in one class's version of the identical methods becoming redundant. For example, a class *"foo_one"* has a method *"draw"* that is responsible for drawing an icon on a screen; another class *"foo_two"* also has a *"draw"* method. Now a designer decides to have a single class *"foo"* and combine the two classes. Instead of having two different "draw" methods the designer can decide to just have one "draw" method.

Assumption 3: the inheritance tree is "full", i.e. there is a root, intermediate nodes and leaves. This assumption merely states that an application does not consist only of standalone classes; there is some use of sub classing.

3. EXISTING METRICS AND ANALYTICAL EVALUATION RESULTS

Inheritance metrics proposed by Chidamber and Kemerer [1], Brito and Carapuca [3], Li's [11] and Rajnish and Bhattacherjee [8] have been summarized in Tables 2, 3, 4 and 5 respectively. The analytical evaluation of Li's inheritance metric suite [11] against Weyuker properties [1] has been presented by Rajnish and Bhattacherjee in [9]. Table 6 summarizes the evaluation results for inheritance metrics proposed by Chidamber and Kemerer, Brito and Carapuca, Li's and Rajnish and Bhattacherjee for Weyuker's Properties. It is immediately observable that Property 9 is not satisfied by any of the metrics listed in Table 5. Failing to meet Property 9, means that complexity cannot be reduced by dividing a class. Since complexity cannot be reduced, it may increase (but also may stay the same).

Table 2: Chidamber and Kemerer Metrics

Metrics	Description
Depth of Inheritance Tree (DIT)	The depth of inheritance will be the maximum length from the node to the root of the tree.
Number of Children (NOC)	Number of immediate subclasses subordinated to a class in the class inheritance tree is the NOC for that class.

Table 3: Li's Inheritance Metrics

Metrics	Description
Number of Ancestor Class (NAC)	The definition of NAC measures the total number of ancestor classes from which a class inherits in the class inheritance tree.
Number of Descendent Class (NDC)	The definition of NDC measures the total number of descendent classes of a class.

Table 4. Brito and Carapuca Inheritance Metrics

Metrics	Description
Total Progeny Count (TPC)	Number of classes that inherit directly or indirectly from a class is the Total Progeny count (TPC) of that class.
Total Parent Count (TPAC)	The number of super classes from which a class inherits directly is the Total Parent Count (TPAC) of that class.
Total Ascendancy Count (TAC)	The number of super classes from which a class inherits directly or indirectly is the Total Ascendancy Count (TAC) of that class.

Table 5. Rajnish and Bhattacherjee Inheritance Metrics

Metrics	Description
(DITC)	Depth of Inheritance Tree of a Class (DITC) metric for class inheritance hierarchy is measured in terms of sum of the attributes (Private, Protected, public and inherited) and Methods (Private, Protected, public and inherited) at each level. The DITC metric of a class is calculated as: $$DITC\ (C_i) = \sum_{i=1}^{L} LEV_i * i$$ Where, LEV_i= Attribute (C_i) + Method (C_i) C_i = A class in the ith level of class inheritance hierarchy. Attribute (C_i) = Count the total number of protected, private, public and inherited attributes within a class in the class inheritance hierarchy at each level. Method (C_i) = Count the total number of protected, private, public and inherited methods within a class in the class inheritance hierarchy at each level. L = Total height in the class inheritance hierarchy i.e. the maximum distance from the last node (last level in the class inheritance hierarchy) to the root node (first level in the class inheritance hierarchy).

Table 6. Analytical Evaluation results.

Property Number	DIT	NOC	TPC	TPAC	TAC	NAC	NDC	DITC
1	√	√	√	√	√	√	√	√
2	√	√	√	√	√	√	√	√
3	√	√	√	√	√	√	√	√
4	√	√	√	√	√	√	√	√
5	×	√	×	√	√	×	√	×
6	√	√	√	√	√	√	√	√
7	√	√	√	√	√	√	√	√
8	√	√	√	√	√	√	√	√
9	×	×	×	×	×	×	×	×

√ Indicates that the metric satisfies the corresponding property.

× Indicates that the metric does not satisfy the corresponding property.

4. POINT OF VIEW OF INHERITANCE

In the April 2001 issue of IEEE Transactions on Software Engineering, a correspondence titled “On the Applicability of Weyuker property 9 to Object-Oriented Structural Inheritance Complexity Metrics” by Gurusaran and Roy was published [4]. In that correspondence Gurusaran and Roy tried to analyze some particular classes of inheritance metrics through formalization and prove that the classes of inheritance metrics can never satisfy property 9 presented by Weyuker [1]. Actually, Gurusaran and Roy’s work was inspired by the observation that none of the inheritance metrics proposed in Chidamber and Kemerer [2] and Brito and Carapuca [3] satisfies property 9 and the arguing for rejection of property 9 for Inheritance Complexity Metrics in Chidamber and Kemerer [2] and Kitchenham et al [10].

Lu Zhang and Dan Xie [7] pointed out some discrepancies in Gurusaran and Roy’s conclusion. Finally, they provided two exceptions to their conclusion.

4.1 Proposed Inheritance Metrics

This section presents proposed inheritance metrics, which are used to calculate the Complexity of an inheritance hierarchy. The primary purpose of the proposed metrics is to evaluate the applicability of Weyuker property 9 to Object-Oriented inheritance tree at the class level as well as at the entire tree level. The new inheritance metrics are defined as follows:

$$ICT(Ci) = \frac{M(Ci) + A(Ci) + IF(Ci)}{N}$$

and

$$ICC(Ci) = M(Ci) + A(Ci) + IF(Ci)$$

$$IF(Ci) = \frac{No\ of\ classes\ inherited\ directly\ by\ Ci}{1 + No\ of\ classes\ inherited\ indirectly\ by\ Ci}$$

Where

Inheritance Complexity of Tree (ICT) is the metric value for the entire tree and inheritance complexity of class (ICC) is the metric value of a class of an inheritance tree.

C_i = Classes at the i^{th} level in an inheritance tree.

A (C_i) = Count the number of attributes (protected, private, public and inherited attributes) at each level in an inheritance tree.

M (C_i) = Count the number of methods (protected, private, public and inherited attributes) at each level in an inheritance tree.

N= Total number of classes in an inheritance tree.

4.2 Examples for Illustration

This section presents three different examples for inheritance tree for the applicability of Weyuker property 9 to an inheritance tree at class level as well as at tree level is presented.

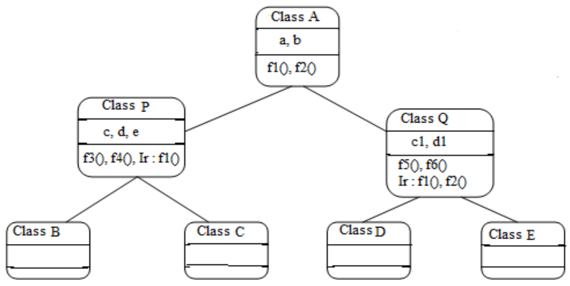

Figure 1a. Class Inheritance Tree when Class P and Q are siblings

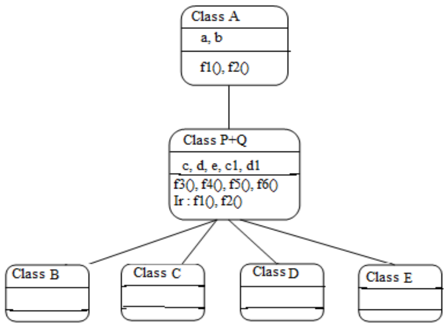

Figure 1b. Class Inheritance Tree when P and Q are combined

Case 1: When class P and Class Q are at the same level in an inheritance tree:

The inheritance metrics for the hierarchies shown in Figures 1a and 1b are calculated as follows:

At Class Level:

ICC (P) =M (P) + A (P) +IF (P)

= 3 + 3 + 2 = 8.

ICC (Q) =M (Q) + A (Q) +IF (Q)

= 4 + 2 + 2 = 8.

ICC (P+Q) = M (P+Q) + A (P+Q) + IF (P+Q)

= 6 + 5 + 4 = 15

From Weyuker Property 9, there exist two classes P and Q such that ICC (P) + ICC (Q) are not < ICC (P+Q). Hence Weyuker Property 9 (Interaction Increases Complexity) is not satisfied.

At Tree Level:

For class P, ICT (P) = 8 / 7 = 1.143.

For class Q, ICT (Q) = 8/ 7 = 1.143.

ICT (P) + ICT (Q) = 1.143 + 1.143 = 2.286.

ICT (P+Q) = 15 / 6 = 2.5.

From Weyuker Property 9, there exist two classes P and Q such that ICT (P) + ICT (Q) < ICC (P+Q). Hence Weyuker Property 9 (Interaction Increases Complexity) is satisfied.

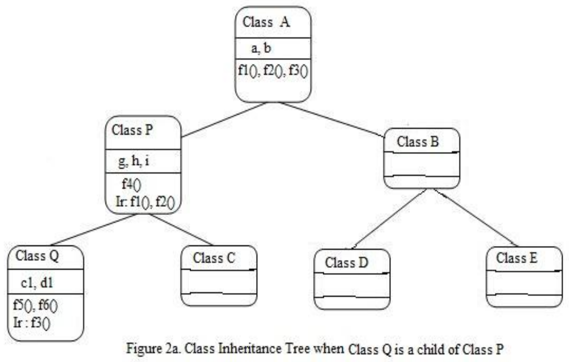

Figure 2a. Class Inheritance Tree when Class Q is a child of Class P

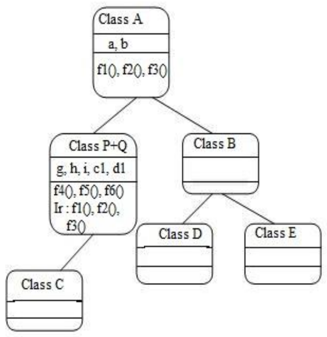

Figure 2b. Class Inheritance Tree when Class P and Class Q are combined

Case 2: When class Q is a child of class P in an inheritance tree:

The inheritance metrics for the hierarchies shown in Figures 2a and 2b are calculated as follows:

At Class Level:

ICC (P) =M (P) + A (P) +IF (P)

= 3 + 3 + 2 = 8.

ICC (Q) =M (Q) + A (Q) +IF (Q)

= 3 + 2 = 5.

ICC (P+Q) = M (P+Q) + A (P+Q) + IF (P+Q)

= 6 + 5 + 1 = 12

From Weyuker Property 9, there exist two classes P and Q such that ICC (P) + ICC (Q) are not < ICC (P+Q). Hence Weyuker Property 9 (Interaction Increases Complexity) is not satisfied.

At Tree Level:

For class P, ICT (P) = 8 / 7 = 1.143

For class Q, ICT (Q) = 5/ 7 = 0.714

ICT (P) + ICT (Q) = 1.143 + 0.714 = 1.857

ICT (P+Q) = 12 / 6 = 2

From Weyuker Property 9, there exist two classes P and Q such that ICT (P) + ICT (Q) < ICC (P+Q). Hence Weyuker Property 9 (Interaction Increases Complexity) is satisfied.

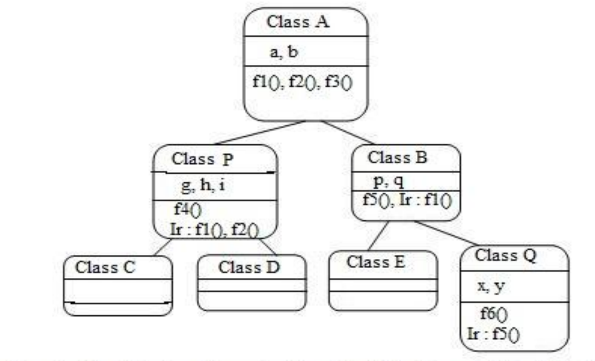

Figure 3a. Class Inheritance Tree when Class P and Class Q are at different Label

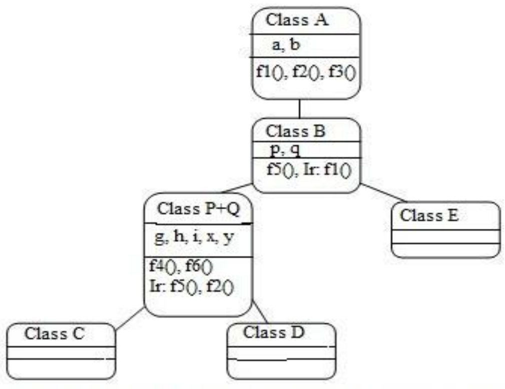

Figure 3b. Class Inheritance Tree when Class P and Class Q are combined

Case 3: When class P and Class Q are not the same level in an inheritance tree and Q is not a child of P:

The inheritance metrics for the hierarchies shown in Figures 3a and 3b are calculated as follows:

At Class Level:

ICC (P) =M (P) + A (P) +IF (P)

= 3 + 3 + 2 = 8.

ICC (Q) =M (Q) + A (Q) +IF (Q)

= 2 + 2 = 4.

ICC (P+Q) = M (P+Q) + A (P+Q) + IF (P+Q)

= 4 + 5 + 2 = 11

From Weyuker Property 9, there exist two classes P and Q such that ICC (P) + ICC (Q) are not < ICC (P+Q). Hence Weyuker Property 9 (Interaction Increases Complexity) is not satisfied.

At Tree Level:

For class P, ICT (P) = 8 / 7 = 1.143

For class Q, ICT (Q) = 4/ 7 = 0.571

ICT (P) + ICT (Q) = 1.143 + 0.571 = 1.714

ICT (P+Q) = 11 / 6 = 1.833

From Weyuker Property 9, there exist two classes P and Q such that ICT (P) + ICT (Q) < ICC (P+Q). Hence Weyuker Property 9 (Interaction Increases Complexity) is satisfied.

From the above example illustration it is observed that the Weyuker property 9 is never satisfied at the class level but at the tree level Weyuker property 9 is satisfied in all the three cases of an inheritance tree.

5. CONCLUSION AND FUTURE SCOPE

It is observed that in all the three cases of an inheritance tree, when two classes are combined, the interaction between classes can increase the complexity of Inheritance Complexity of Tree (ICT) value (see Section 4.2). Hence, Weyuker property 9 (Interaction Increases Complexity) is satisfied in all the three cases of an inheritance tree. Satisfying property 9 indicate that the complexity of Object-Oriented programs may increase when number of classes reduced.

The future scope includes the investigation of other metrics for combination of classes and validation on industrial data.

6. ACKNOWLEDGMENTS

We would like to thanks BIT, MESRA and specially Department of Computer Science & Engineering for encouraging us to do this work.

7. REFERENCES

[1] Weyuker E. J., “Evaluating Software Complexity Measures”, IEEE Trans. on Software Engineering, 14, (1998), pp.1357-1365.

[2] Chidamber S. R. and Kemerer C. F., “A Metrics Suite for Object Oriented Design”, IEEE Trans. on Software Engineering, 20, 6(1994), pp.476 – 493.

[3] Brito A.F. and Carapuca R., “Candidate Metrics for Object-Oriented Software within a Taxonomy Framework”, Journal of System Software, vol. 26, 1994, pp. 87-96.

[4] Gursaran and Roy G, “On the applicability of Weyuker Property Nine to Object-Oriented Structural Inheritance Complexity Metrics”, IEEE Transaction on Software Engineering, Vol.27, no.4, 2001, pp. 361-364.

[5] Cherniavsky J. and.Smith C.” On Weyuker’s Axioms for Software Complexity Measures”, IEEE Transaction on Software Engineering, Vol. 17, no-6, 1991, pp. 636-638.

[6] Sharma N., Joshi P., Joshi R.K. “Applicability of Weyuker’s Property 9 to Object-Oriented Metrics”, IEEE Transaction on Software Engineering, Vol.32, No.3, 2006, pp. 209-211.

[7] Zhang L. and Xie D., “Comments on ‘On the applicability of Weyuker Property Nine to Object-Oriented Structural Inheritance Complexity Metrics”, IEEE Transaction on Software Engineering, Vol.28, no.5, 2002, pp. 526-527.

[8] Rajnish K. and Bhattacherjee V., “Class Inheritance Metrics-An Analytical and Empirical Approach”, Infocomp Journal of Computer Science, Federal University of Lavras, Brazil, Vol. 7 No.3, pp. 25-34, 2008.

[9] Rajnish K. and Bhattacherjee V.,” An Analytical Evaluation on Li's Inheritance Metric Suites against Weyuker's Properties”, International Journal of Engineering and Technology Volume 1 No. 1, October, 2011, PP: 80-88.

[10] Kitchenham B, Pfleeger Sl., Fenton NE., “Towards a framework for software measurement validation”, IEEE Trans. On Software Engineering 1995; 21(12):929-944.

[11] Li W. Another metric suite for object-oriented programming. The Journal of Systems and Software 1998; 44(2):155–162.

[12] Parther R.E., "An Axiomatic Theory of Software Complexity Measurement", Computing Journal, vol.27, No. 4, 1984, PP: 340-346.

[13] Melton A., Gustafson D., Bieman J., and Baker A., "A Mathematical Perspective for Software Measure Research", Journal of Software Engg. Vol. 5, No. 5, 1990 PP: 246-254.

[14] Roy G., "On the Applicability of Weyuker Property Nine to Object-Oriented Structural Inheritance Complexity Metrics, M. Tech. Minor Project Report, Faculty of Eng,. Dayalbagh Educational Inst., Agra, 1997.

[15] Rajnish K. and Bhattacherjee V. "Maintenance of Metrics through class Inheritance hierarchy", Proceedings of International conference on Challenges and Opportunities in Information Technology Industry", PCTE, Ludhiana, 2005, pp.83.

[16] Rajnish K. and Bhattacherjee V. "A New Metric for Class Inheritance Hierarchy: An Illustration", Proceedings of National Conference on Emerging Principles and Practices of Computer Science & Information Technology (EPPCSIT'06)", GNDEC, Ludhiana, Allied Publishers, 2006, pp.321-325,

[17] Rajnish K. and Bhattacherjee V. "Class Inheritance Metrics and development Time: A Study", International Journal Titled as "PCTE Journal of Computer Science, Vol.2, Issue 2, July-Dec-06, pp. 22-28.

[18] Fenton N., "Software Measurement: A necessary scientific basis", IEEE Transaction on Software Engineering, Vol.20, No.6, 1994, pp.199-206.