Effectiveness of Object Oriented Inheritance Metrics in Software Reusability

Muhammad Ilyas[1(✉)], Josef Küng[2], and Van Quoc Phuong Huynh[2]

[1] Department of Computer Science and IT,
University of Sargodha, Sargodha, Pakistan
m.ilyas@uos.edu.pk

[2] FAW, Johannes Kepler University Linz, 4040 Linz, Austria
{Jkueng, vqphuynh}@faw.jku.at

Abstract. Inheritance is a key feature of object oriented paradigm. It is actually the sharing of attributes and operations among classes based on a hierarchical relationship. Software reusability is the basic concept of software engineering that is affected by the sophistication of inheritance hierarchy so in order to determine complexity of inheritance which in turn has impact on software reusability; we have proposed class inheritance metrics and explained them in an elaborative manner. In the work presented here we proposed different class inheritance metrics, compared them with existing ones and attempted to present an alternate solution with some extended features to find out intricacy of class inheritance which significantly concerns with reusability.

Keywords: Hierarchy · Inheritance · Metrics · Object oriented · Reusability

1 Introduction

Software reusability is an imperative requirement for cost and time optimized software development. Inheritance is considered as one of the most vital features of OO paradigm which has great impact on software reusability. Intricacy of inheritance hierarchy greatly affects the reusability as well as maintainability of system so it is important for designers to know about the complexity of inheritance hierarchy. Inheritance metrics actually provides information regarding complexity of inheritance that's why we are focusing on inheritance metrics with respect to software reusability here [13, 15]. Research work in [8] also focuses on fruitfulness of reuse as it emphasize on pattern's analysis like a way to assist the reuse of software knowledge by confining conceptual models in those domains. Reuse of software through inheritance fabricates more comprehensible, maintainable and trustworthy software [2]. Research work in [3, 6, 7] specifies that if a system is not utilizing inheritance then that would be better in maintainability as compared with a system utilizing inheritance.

Nevertheless, this fact is approved that deeper inheritance hierarchy leads to better reusability and harder maintainability of the system. So designers possibly will be liable to maintain the inheritance hierarchy trivial, dumping reusability through inheritance for the ease of comprehension [5]. Thus it is essential to determine the intricacy of inheritance hierarchy for the purpose of determining the discrepancies

T.K. Dang et al. (Eds.): FDSE 2017, LNCS 10646, pp. 231–239, 2017.
https://doi.org/10.1007/978-3-319-70004-5_16

between the shallowness and depth of inheritance hierarchy. In this paper we consider the inheritance metrics presented in [10] and [11] for comparison as these metrics give information about class inheritance hierarchy and reusability in much comprehensible way. During their study we came to know that these metrics give information about complexity of inheritance hierarchy in terms of subclasses/children but what will be intricacy of inheritance hierarchy w.r.t. parent classes/super classes which in turn helps us in determining the reusability. So in order to explore this phenomenon we have proposed our own inheritance metrics.

Rest of the paper is organized is as follows. Section 2 shows related work and Sect. 3 presents proposed metric suite of inheritance. In Sect. 4 there are two examples of inheritance trees and their calculations to illustrate existing and proposed metrics. Section 5 presents results of existing and proposed inheritance metrics and Sect. 6 explains analysis of results of existing and proposed inheritance metrics. Section 7 presents conclusion and future works respectively.

2 Related Work

Software metrics have indispensable role in the field of software engineering for the purpose of determining and approximating software quality, complexity, reusability, cost and project exertion etc. Here we will give brief overview of OO based metrics proposed by researchers and then we give emphasis of inheritance metrics.

In **Existing Object Oriented Metrics** several OO based metrics have been proposed till now for the measurement and illustration of different aspects of OO paradigm e.g. cohesion, inheritance and encapsulation etc. Metrics suite proposed in [4, 5] is one of the best known OO metrics suite relating to inheritance, cohesion and coupling etc. There are five metrics defined for measuring a component's comprehension, adaptability and portability with confidence intervals [14]. Another set of metrics is considered suitable for evaluating the use of the main abstractions of the OO paradigm such as inheritance, encapsulation, information hiding or polymorphism [1]. Authors [9] have surveyed metrics proposed for OO systems and mainly focused on product metrics that can be applied to an advanced design. Inheritance metrics proposed in described the inheritance of inheritance tree in OOD as Average Degree of Understand-ability (AU) and Average Degree of Modifiability (AM) [12].

In **Existing Inheritance Metrics,** there are number of authors who have proposed different inheritance metrics but here we consider inheritance metrics of [10, 11] as these metrics give information about class inheritance hierarchy and reusability in much comprehensible and graspable way.

In **Average Inheritance Depth (AID)** Metrics, it is suggested in [10] that AID metric is the mean depth of inheritance tree and this is an extension of [4, 5]'s DIT (Depth of Inheritance tree) metric. The AID of a class is calculated by [11] as:

$$\text{AID} = \sum (\text{Depth of each class}) / \textit{Number of Classes} \tag{1}$$

This metric AID is actually the ratio of sum of depth of each class in class inheritance tree to the total number of classes in that inheritance tree. Different inheritance metrics are presented in [10] as DBRM, ANDC and ANIC.

Derived Base Ratio (DBR). Metric is the ratio of the total derived classes to the total base classes in the class inheritance Tree [10]. DBRM is calculated as follows:

$$\mathrm{DBRM} = \sum_{\mathrm{i}=0}^{\mathrm{N}} \mathrm{TD}(\mathrm{C_i}) / \sum_{\mathrm{i}=0}^{\mathrm{N}} \mathrm{TB}(\mathrm{C_i}) \tag{2}$$

N: Total number of classes in the class inheritance tree. For rest of all equations, N represents the same.

Average Number of Direct Child (ANDC). Metric is the ratio of the total number of immediate child to the total number of classes in the inheritance tree [10]. ANDC metric is calculated as follows:

$$\mathrm{ANDC} = \sum_{\mathrm{i}=0}^{\mathrm{N}} \mathrm{TDC}(\mathrm{C_i}) / N \tag{3}$$

Average Number of Indirect Child (ANIC). Metric is the ratio of the total number of indirect child to the total number of classes in the inheritance tree [10]. ANIC metric is calculated as follows:

$$\mathrm{ANIC} = \sum_{\mathrm{i}=0}^{\mathrm{N}} \mathrm{TIC}(\mathrm{C_i}) / N \tag{4}$$

3 Proposed Inheritance Metrics

We have proposed four inheritance metrics which will give information about complexity of class inheritance hierarchy in terms of super classes/parent classes, ancestors and descendants. These are named as Average Number of Direct Parents (ANDP) metric, Average Number of Indirect Parents (ANIP) metric, Average Number of Ancestor Classes (ANAC) metric and Average Number of Descendant Classes (AND) metric. These metrics with formulas and assumptions are explained as follows.

Average Number of Direct Parents (ANDP) Metric is the ratio of the total number of direct or immediate parents to the total number of classes in the inheritance tree. Formula to calculate ANDP metric along with assumption is given as follows:

$$\mathrm{ANDP} = \sum_{\mathrm{i}=0}^{\mathrm{N}} \mathrm{TDP}(\mathrm{C_i}) / N \tag{5}$$

- ANDP metric gives information about number of direct parents/direct super classes in class inheritance tree.
- ANDP metric measures that from how many direct super classes; subclasses are going to inherit the properties such as methods and attributes.

Average Number of Indirect Parents (ANIP) Metric is the ratio of the total number of indirect parents to the total number of classes in the inheritance tree. Equation 6 has the formula to calculate ANIP metric and assumption are as follows:

$$\text{ANIP} = \sum_{i=0}^{N} \text{TIP}(C_i)/N \tag{6}$$

- ANIP metric gives us indication of how many indirect parents/indirect super classes are going to affect subclasses in class inheritance tree.
- Deepness of a class in class inheritance tree shows higher degree of inheritance, thus it can be hard to comprehend a system with many inheritance layers.
- Greater value of ANIP metric indicates that many methods might be reused.

Average Number of Ancestor Classes (ANAC) Metric is the ratio of the total number of ancestor classes to the total number of classes in the inheritance tree. ANAC metric is calculated with formula of Eq. 7. Later on are assumptions.

$$\text{ANAC} = \sum_{i=0}^{N} \text{TA}(C_i)/N \tag{7}$$

- ANAC metric gives information about how many ancestor classes are present in class inheritance tree.
- Greater value of ANAC of class inheritance tree gives the indication that tree has deeper inheritance hierarchy than a class inheritance tree having low value of ANAC.
- Class Inheritance tree having shallow inheritance hierarchy constitute lesser design complexity, since less methods and classes are involved.

Average Number of Descendants Classes (AND) Metric is the ratio of the total number of descendant classes to the total number of classes in the inheritance tree. Equation 8 is showing AND metric calculation formula.

$$\text{AND} = \sum_{i=0}^{N} \text{TD}(C_i)/N \tag{8}$$

Assumptions of AND Metrics are as following:

- AND metric gives information about how many descendant classes are present in class inheritance tree.
- Low value of AND of class inheritance tree gives the indication that tree has shallow inheritance hierarchy than a class inheritance tree having high value of AND.
- AND metrics provides information about inheritance and inheritance plays important role in reuse of already designed classes when designing a new class.

4 Examples for Illustration

In order to explain and illustrate these proposed inheritance metrics and to compare them with existing ones we have taken two class inheritance trees. In these class inheritance trees rectangular boxes represents classes and arrow symbol show relationship between these classes.

In first example (See Fig. 1) there is a simple class inheritance tree having 6 classes. Class A is root class and then downward there are descendants of root class A. As class B and Class C are child classes of Class A. Class D and Class E are child classes of Class C similarly Class F is child class of Class B. It is quite obvious from Fig. 1 that inheritance tree is simple in structure as it has little depth and breadth.

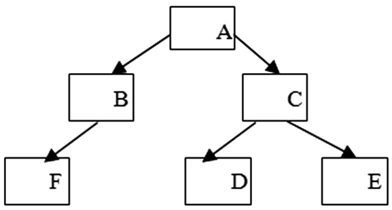

Fig. 1. Class inheritance tree having less depth and breadth

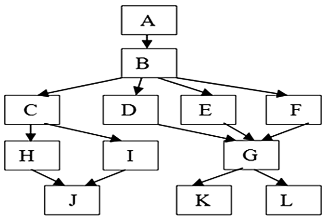

Fig. 2. Class inheritance tree having more depth and breadth

In second example (See Fig. 2) we have taken a little complex class inheritance tree having 12 classes. Class A is root class and then downward there are descendants of root class A. As Class B is child class of Class A. Classes C, D, E and F are child classes of Class B. Classes H and I are child classes of class C and Class H and I are parents of Class J. Similarly Classes D, E and F are parents of Class G and Class G is Parent Class for Classes K and L. Figure of this 2nd class inheritance tree shows that this inheritance tree is bit complex in structure as it has more depth and breadth as compared to class inheritance tree of 1st example.

Table 1. Existing and proposed metrics calculations for inheritance tree of Fig. 1

Calculation of AID: AID is calculated using (1) as follows:
depth of class A = 1; depth of class B = 2; depth of class C = 2; depth of class D = 3; depth of class E = 3; depth of class F = 3; AID = 14/6 = 2.33
Calculation of DBRM: DBRM is calculated using (2) as follows:
Total base classes = 3; Total derived classes = 5; DBRM = 5/3 = 1.66
Calculation of ANDC: ANDC is calculated using (3) as follows:
TDC(A) = 2; TDC(B) = 1; TDC(C) = 2; TDC (D) = 0; TDC (E) = 0; TDC(F) = 0; ANDC = 5/6 = 0.833
Calculation of ANIC: ANIC is calculated using (4) as follows:
TIC (A) = 3; TIC (B) = 0; TIC (C) = 0; TIC (D) = 0; TIC (E) = 0; TIC (F) = 0; ANIC = 3/6 = 0.5
Calculation of ANDP: ANDP is calculated using (5) as follows:
TDP (A) = 0; TDP(B) = 1; TDP(C) = 1; TDP(D) = 1; TDP (E) = 1; TDP (F) = 1; ANIC = 5/6 = 0.833
Calculation of ANIP: ANIP is calculated using (6) as follows:
TIP (A) = 0; TIP (B) = 0; TIP(C) = 0; TIP (D) = 1; TIP (E) = 1; TIP (F) = 1; ANIC = 3/6 = 0.5
Calculation of ANAC: ANAC is calculated using (7) as follows:
TA (A) = 0; TA (B) = 1; TA(C) = 1; TA (D) = 2; TA (E) = 2; TA (F) = 2; ANIC = 8/6 = 1.33
Calculation of AND: AND is calculated using (8) as follows:
TD (A) = 5; TD (B) = 1; TD(C) = 2; TD (D) = 0; TD (E) = 0; TD (F) = 0; AND = 8/6 = 1.33

5 Results of Existing and Proposed Metrics

In Tables 1 and 2, we have presented the calculation of existing and proposed inheritance metrics for class inheritance tree of example 1 and example 2 respectively. The entire results of existing and proposed inheritance metrics for both class inheritance trees are shown in Table 3 and also in Fig. 3 in graphical form.

Results of existing and proposed inheritance metrics values for both class inheritance trees can be shown graphically as in Fig. 3 as both inheritance trees are represented here on x-axis and metric values on y-axis.

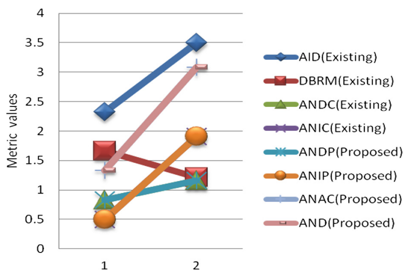

Fig. 3. Graphical representation of result of existing and proposed inheritance metrics for both class inheritance trees of Figs. 1 and 2.

Table 2. Existing and proposed metrics calculations for inheritance tree of Fig. 2

Calculation of AID: AID is calculated using Eq. (1) as follows:
depth of class A = 1; depth of class B = 2; depth of class C = 3;
depth of class D = 3; depth of class E = 3; depth of class F = 3;
depth of class G = 4; depth of class H = 4; depth of class I = 4;
depth of class J = 5; depth of class K = 5; depth of class L = 5; AID = 42/12 = 3.5
Calculation of DBRM: DBRM is calculated using Eq. (2) as follows:
Total base classes = 9; Total derived classes = 11; DBRM = 11/9 = 1.22
Calculation of ANDC: ANDC is calculated using Eq. (3) as follows:
TDC (A) = 1; TDC (B) = 4; TDC (C) = 2; TDC (D) = 1; TDC (E) = 1; TDC (F) = 1; TDC (G) = 2; TDC (H) = 1; TDC (I) = 1; TDC (J) = 0; TDC (K) = 0; TDC (L) = 0;
ANDC = 14/12 = 1.16
Calculation of ANIC: ANIC is calculated using Eq. (4) as follows:
TIC (A) = 10; TIC (B) = 6; TIC (C) = 1; TIC (D) = 2; TIC (E) = 2; TIC (F) = 2; TIC (G) = 0; TIC (H) = 0; TIC (I) = 0; TIC (J) = 0; TIC (K) = 0; TIC (L) = 0; ANIC = 23/12 = 1.91
Calculation of ANDP: ANDP is calculated using Eq. (5) as follows:
TDP (A) = 0; TDP (B) = 1; TDP(C) = 1; TDP (D) = 1; TDP (E) = 1; TDP (F) = 1; TDP (G) = 3; TDP (H) = 1; TDP (I) = 1; TDP (J) = 2; TDP (K) = 1; TDP (L) = 1;
ANIC = 14/12 = 1.16
Calculation of ANIP: ANIP is calculated using Eq. (6) as follows:
TIP (A) = 0; TIP (B) = 0; TIP(C) = 1; TIP (D) = 1; TIP (E) = 1; TIP (F) = 1; TIP (G) = 2; TIP (H) = 2; TIP (I) = 2; TIP (J) = 3; TIP (K) = 5; TIP (L) = 5; ANIC = 23/12 = 1.91
Calculation of ANAC: ANAC is calculated using Eq. (7) as follows:
TA (A) = 0; TA (B) = 1; TA(C) = 2; TA (D) = 2; TA (E) = 2; TA (F) = 2; TA (G) = 5; TA (H) = 3;
TA (I) = 3; TA (J) = 5; TA (K) = 6; TA (L) = 6; ANAC = 37/12 = 3.08
Calculation of AND: AND is calculated using Eq. (8) as follows:
TD (A) = 11; TD (B) = 10; TD(C) = 3; TD (D) = 3; TD (E) = 3; TD (F) = 3; TD (G) = 2; TD (H) = 1; TD (I) = 1; TD (J) = 0; TD (K) = 0; TD (L) = 0; AND = 37/12 = 3.08

6 Analysis of Results

From above mentioned Table 3 and graph of Fig. 3, we have certain observations regarding existing and proposed inheritance metrics with respect to reusability.

- DBRM metric value of inheritance tree of Fig. 2 is lower than that of inheritance tree of Fig. 1, means Fig. 2 has low ratio of derived classes to base classes.
- AID Metric is concerned with depth of inheritance tree so lower AID metric value indicates low design complexity, which leads to easy maintainability and more understand-ability but low reusability.
- Similarly from Figs. 1, 2 and 3 and Table 3, it can be analyzed that inheritance in Fig. 2 reuse more properties (ANDC and ANIC Metric value) means better reusability as inheritance of Fig. 2 has high value of ANDC and ANIC metrics.
- Figure 2 has greater ANIP metric value as compared to inheritance tree of Fig. 1 means class inheritance in Fig. 2 is deeper. Deeper tree constitute greater design complexity, as more classes are involved. We can say that higher ANIP metric value leads to more reusability but lower maintainability and understandability.

- Table 1 and Fig. 3 shows that ANDC and ANDP metrics values are similar. It indicates that any of these metric can be used to find out overall depth of inheritance of inheritance tree which will determine the reusability of that component.
- From Table 3 it can be analyzed that value of ANIC metric (Existing) and ANIP metric (proposed) for each inheritance tree is similar. So it also gives us a facility to use both these metrics alternatively. Same is indicated by graph given above.
- Both (proposed metrics) ANAC and AND Metric values indicate the depth of inheritance of class inheritance tree. It can be easily analyzed that class inheritance tree of Fig. 2 has more values of ANAC and AND Metric, which shows Fig. 2's tree has deeper inheritance hierarchy and more design complexity and more reusability than class inheritance tree of Fig. 1.

Table 3. Result of existing and proposed inheritance metrics for both class inheritance trees of Figs. 1 and 2

Inheritance metrics	Values for Fig. 1	Values for Fig. 2
AID(Existing)	2.33	3.5
DBRM(Existing)	1.66	1.22
ANDC(Existing)	0.833	1.16
ANIC(Existing)	0.5	1.91
ANDP(Proposed)	0.833	1.16
ANIP (Proposed)	0.5	1.91
ANAC (Proposed)	1.33	3.08
AND (Proposed)	1.33	3.08

7 Conclusion and Future Works

Class design plays vital role in OO system's development and inheritance supports the class hierarchy design. Reusability is related to depth and shallowness of inheritance hierarchy so it is essential to determine the intricacy of inheritance hierarchy. We have proposed different class inheritance metrics to give information about inheritance complexity with respect to parent classes, ancestor and descendant classes.

Two of our proposed metrics ANDP and ANIP tend to provide an alternate solution to discover intricacy of class inheritance as compared with existing inheritance metrics ANDC and ANIC because our proposed ones provide same results in terms of super classes so we come to know that intricacy of inheritance hierarchy is same w.r.t parent classes as that of subclasses. We have proposed two more inheritance metrics as AND metric and ANAC metric which can be used interchangeably according to ease and given conditions. These inheritance metrics are involved in finding intricacy of class inheritance hierarchy that has dominant effects on software reusability.

In future, this work can be extended for deeper and complicated inheritance trees because class inheritance trees studied here are small as compared to inheritance trees used usually in systems so that effect of these proposed inheritance metrics on reusability process can be evaluated more precisely and effectively.

References

1. Abreu, F.B., Carapuça, R.: Object-oriented software engineering: measuring and controlling the development process. In: 4th International Conference on Software Quality, USA, October 1994
2. Basili, V.R., Briand, L.C., Melo, W.L.: A validation of object-oriented design metrics as quality indicators, Technical report, University of Maryland, pp. 1–24 (1995)
3. Poornima, U.S., Suma, V.: Impact of multiple inheritances on cohesion complexity in software design. In: ICICT, Coimbatore, pp. 1–4 (2016)
4. Chidamber, S.R., Kemerer, C.F.: Towards a metric suite for object-oriented design. In: Proceedings of the Sixth OOPSLA Conference, pp. 197–211 (1991)
5. Chidamber, S.R., Kemerer, C.F.: A metric suite for object-oriented design. IEEE Trans. Softw. Eng. **20**, 476–493 (1994)
6. Daly, J., Brooks, A., Miller, J., Roper, M., Wood, M.: Evaluation inheritance depth on the maintainability of object-oriented software. Empirical Softw. Eng. **1**, 109–132 (1996)
7. Harrison, R., Counsell, S.J., Nithi, R.V.: An evaluation of the MOOD set of object-oriented software metrics. IEEE Trans. SE **24**(6), 491–496 (1998)
8. Jawawi, D., Deris, S., Mamat, R.: Software reuse for mobile robot applications through analysis patterns. IAJIT **4**(3), 220–228 (2007)
9. Purao, S., Vaishnavi, V.: Product metrics for object-oriented systems. ACM Comput. Surv. **35**(2), 191–221 (2003)
10. Rajnish, K., Choudhary, A.K., Agrawal, A.M.: Inheritance metrics for object-oriented design. IJCSIT **2**(6), 13–26 (2010)
11. Seller, B.H.: Object-Oriented Metrics: Measures of Complexity. Prentice Hall PTR, Englewood Cliffs (1996)
12. Sheldon, T.F., Jerath, K., Chung, H.: Metrics for maintainability of class inheritance hierarchies. J. Softw. Maintenance Evol. Res. Pract. **14**, 1–14 (2002)
13. Singh, S., Thapa, M., Singh, G.: Software engineering survey of reusability based on software component. IJCSIT **2**(6) (2010)
14. Washizaki, H., Yamamoto, H., Fukazawa, Y.: A metrics suite for measuring reusability of software components. In: Software Metrics Symposium, pp. 211–223, September 2003
15. Ilyas, M., Abbas, M., Saleem, K.: A metric based approach to extract, store and deploy software reusable components effectively. IJCSI **10**, 257–264 (2013)