Developing a Nested Class Complexity Metric for Nested Classes

Rajender Singh Chhillar
M.D. University, Rohtak, Haryana, India
Email: chhillar02@gmail.com

Parveen Kajla and Usha Chhillar
M.D. University, Rohtak and AIJHM (PG) College, Rohtak,Haryana,India
Email: {pkajla77, chhillarusha01}@gmail.com

***Abstract*—In Object-oriented programming languages like Java; it is the basic need to define a class within another class. These classes are known as nested classes or inner classes. The scope of a nested class is limited to its outer class. All the variables and methods of outer class are accessible inside inner class enhances encapsulation. Nested classes also help in packaging of the classes. In this paper, we propose a new metric, namely, Nested Class Complexity Metric (NCCM) to measure the complexity of nested classes and the results are compared with existing metrics, which are quite encouraging.**

***Index Terms*—nested classes, complexity metrics, NCCM, packaging, encapsulation**

I. INTRODUCTION

The object oriented approach consists of two basic terms Class and Object. A class is a blueprint or prototype that defines the variables and the methods common to all objects of a certain kind. The main difference between a class and an object is that objects are tangible, but a class is always intangible. Classes provide the benefit of reusability. A number of Metrics have been proposed in object-oriented programming for classes, inheritance, coupling, cohesion, and polymorphism [1]-[6].

Inheritance provides a very helpful concept of hierarchy and code reusability. Most of the object oriented languages implement the concept of Nested Classes or inner classes i.e. class within a class. Inner classes share all the features of a regular class. They could contain constructors, attributes, methods and further inner classes.

This nested feature reduces coupling and increases cohesion of the system which is desirable but on the other hand excessive use, affects the readability of the system and thus increases the complexity and maintainability of the system [7].

Nested classes are the basic needs in the languages like Java. These languages also support Nested Methods or calling of a method into the methods of the same class. Thus, if class B is defined within class A, then B is known to A, but not outside of A. A nested class has access to the members, including private members, of the class in which it is nested. However, the enclosing class does not have access to the members of the nested class.

A static nested class is one which has the static modifier applied. Because it is static, it must access the members of its enclosing class through an object. That is, it cannot refer to members of its enclosing class directly. Because of this restriction, static nested classes are seldom used.

The most important type of nested class is the inner class. An inner class is a non-static nested class. It has access to all of the variables and methods of its outer class and may refer to them directly in the same way that other non-static members of the outer class do. Thus, an inner class is fully within the scope of its enclosing class.

Most of the researchers focus on object oriented metrics [8]-[12] and its complexity [13]-[17] and a few on inner classes [18] and [19]. In this paper, a new metric NCCM is proposed to check the nested behavior of the classes. Fig. 1 shows a basic program implements classes within a class. It defines a class Saving Account and class Current Account within the class Bank Account.

A Sample Program for Nested Classes

```
class BankAccount
{
BankAccount();

class SavingAccount
   {
SavingAccount();
   };

class CurrentAccount
   {
CurrentAccount();
   };

};
```

Figure 1. Nested class example.

This paper comprises of five sections. Section II depicts the new Nested Classes Complexity Metric (NCCM) for object oriented software development.

Manuscript received June 27, 2013; revised September 11, 2013.

doi: 10.12720/ijoee.1.4.244-248

Section III illustrates the experimental results of proposed metric. Section IV compares the results of proposed metric with existing metrics. Section V refers concluding remarks and future scope.

II. PROPOSED NESTED CLASSES COMPLEXITY METRIC

Software, designed using object oriented approach consists of classes and within that data members and member functions. Considering the above program, it is observed that the readability, complexity and maintainability of the software in object oriented approach are not only depending upon the number of classes (nC) but also on their level of existence (L) in their structure. The excessive use of nested classes increases the difficulty level during maintainability.

In this study, we consider the root level (L=0) as outer class and thus consider the nested classes from the first level (L=1). We can define the Complexity Metric (NCCM) at each nested level as

$$\mathrm{NCCM(l)} = \frac{1}{\mathrm{L} + (\text{total no. of nested classes})}$$

Thus to count the number of classes in a program at first level (where L=1)

$$nCl = C1 + C2 + C3 + \cdots + Cnl$$

$$nCl = \sum_{il=1}^{nl} Cil$$

$$OR \quad nC1 = \sum_{i1=1}^{n1} Ci1 \quad , \text{where } \mathrm{l} = 1$$

Similarly for second nested level is

$$nCl = \sum_{il=1}^{nl} Cil$$

$$OR \quad nC2 = \sum_{i2=1}^{n2} Ci2 \quad , \text{where } \mathrm{l} = 2$$

And so on for pth nested level is

$$nCp = \sum_{ik=1}^{nk} Cik \quad , \quad , where\ l = k$$

Using the above equations, we can define the Complexity Metric (NCCM) as

$$NCCM = \sum_{t=1}^{k} \left(\frac{1}{t + \sum_{it=1}^{nt} Ct_{it}} \right)$$

where

k is the total number of nested levels

n is the total number of classes (C) at each level

Sample programs are shown in Fig. 2 and Fig. 3, which shows class hierarchy having five classes with two and five nested level respectively. NCCM value for these programs are calculated and which shows that Fig. 2 have less complexity than Fig. 3.

```
class A
{
  Class B { }
  Class C { }
  Class D { }
  Class E { }
}
```

Nested Levels = 2

No. of immediate nested classes at Level 1= 4

NCCM(1) = $\frac{1}{1+4}$ = 0.20

No. of immediate nested classes at Level 2=0

NCCM(2) = $\frac{1}{2+0}$ = 0.50

NCCM = $\sum_{x=1}^{2} NCCM(x)$
= 0.20+0.50=0.70

Figure 2. NCCM value=0.70.

```
class A
{
  Class B
  {
    Class C
    {
      Class D
      {
        Class E { }
      }
    }
  }
}
```

Nested Levels = 5

No. of immediate nested classes at Level 1= 1

NCCM(1) = $\frac{1}{1+1}$ = 0.50

No. of immediate nested classes at Level 2= 1

NCCM(2) = $\frac{1}{2+1}$ = 0.33

No. of immediate nested classes at Level 3= 1

NCCM(3) = $\frac{1}{3+1}$ = 0.25

No. of immediate nested classes at Level 4= 1

NCCM(4) = $\frac{1}{4+1}$ = 0.20

No. of immediate nested classes at Level 5= 0

NCCM(5) = $\frac{1}{5+0}$ = 0.20

NCCM = $\sum_{x=1}^{5} NCCM(x)$
= 0.50+0.33+0.25+0.20+0.20
=1.48

Figure 3. NCCM value=1.48.

III. EXPERIMENTAL RESULTS

In order to measure the maintainability, complexity of Nested Classes in Object oriented systems, first of all the 15 programs are developed using object oriented language java; with two, three, four and five classes and then the proposed metric is applied.

The programs P1 to P15 are arranged in such a manner that they are sorted by number of nested classes and then by number of immediate inner classes to calculate the NCCM value by implementing the proposed metric in

Table I. Program P1 has minimum level of immediate inner classes, whereas program P15 has maximum level of immediate inner classes.

TABLE I. NCCM VALUE FOR PROGRAM P1 TO P15

Programs	Number of Nested Classes	Number of immediate inner classes at					NCCM Value
		Level-1	Level-2	Level-3	Level-4	Level-5	
P1	1	1	0	--	--	--	1.00
P2	2	2	0	--	--	--	0.83
P3	2	1	1	0	--	--	1.17
P4	3	3	0	--	--	--	0.75
P5	3	2	1	0	--	--	1.00
P6	3	1	2	0	--	--	1.08
P7	3	1	1	1	0	--	1.33
P8	4	4	0	--	--	--	0.70
P9	4	2	2	0	--	--	0.92
P10	4	3	1	0	--	--	0.92
P11	4	1	3	0	--	--	1.03
P12	4	2	1	1	0	--	1.17
P13	4	1	2	1	0	--	1.25
P14	4	1	1	2	0	--	1.28
P15	4	1	1	1	1	0	1.48

IV. COMPARISION WITH EXISTING METRICS

The results of the proposed metric are compared with the existing metrics proposed by various researchers. Existing Metrics like Depth Inheritance Tree (DIT) [7], Maintainability Metric (M) [18] and Complexity Metric(C) [19] for inner classes are used to compare the results.

DIT = the maximum length from the node to the root of the tree.

$$M = \sum \frac{1}{1+n}$$

where n denotes number of immediate inner classes of an outer class

$$C = \sum \frac{b}{d}$$

where b denotes the breadth of a particular depth level and d denotes the depth level

A. *Comparison of Metrics with Three Classes (Two Nested)*

TABLE II. COMPARISON OF NCCM VALUE OF PROGRAM P2 TO P3

Program	Number of Nested Classes	DIT	NCCM	M	C
P2	2	1	0.83	2.33	2.00
P3	2	2	1.17	2.00	1.83

Program P2 and P3 in Table II shows that, the value of NCCM is in increasing order and the value of M& C are in decreasing order. With the increase in DIT from 1 to 2 the complexity increases. The value of M=2.33 with DIT=1 is more than M=2.0 with DIT=2. Similarly the value C=2.0 with DIT=1 is more than C=1.83 with DIT=2. But with NCCM, the value increases with the increase of DIT. The graph in Fig. 4 shows negative slope with M & C and which is inverse to DIT and NCCM having positive slope.

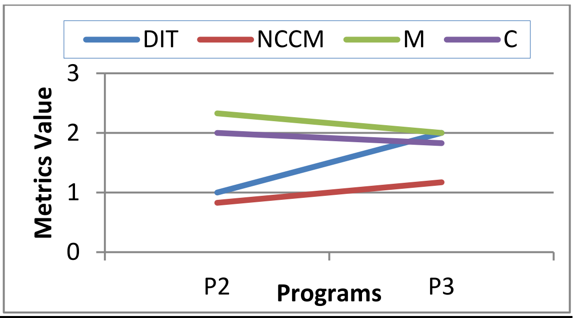

Figure 4. Metrics value of program P2 to P3.

B. *Comparison of Metrics with Four Classes (Three Nested)*

TABLE III. COMPARISON OF NCCM VALUE OF PROGRAM P4 TO P7

Program	Number of Nested Classes	DIT	NCCM	M	C
P4	3	1	0.75	3.25	2.50
P5	3	2	1.00	2.83	2.33
P6	3	2	1.08	2.83	2.17
P7	3	3	1.33	2.50	2.08

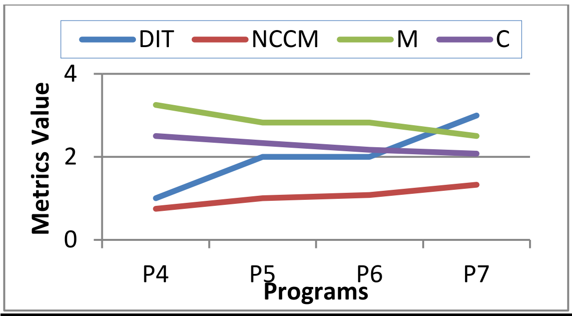

Figure 5. Metrics value of program P4 to P7.

Program P4, P5, P6 and P7 in Table III shows that, the value of NCCM is in increasing order and the value of M & C are in decreasing order. With the increase in DIT from 1 to 3 the complexity increases. The value of M=3.25 with DIT=1 is much more than M=2.50 with DIT=3. Similarly the value of C=2.50 with DIT=1 is more that C=2.08 with DIT=3. But with NCCM, the value increases from 0.75 to 1.33 with the increase of DIT from 1 to 3. The graph in Fig. 5 shows negative slope with M & C and which is inverse to DIT and NCCM having positive slope.

C. *Comparison of Metrics with Five Classes (Four Nested)*

Program P8, P9, P10, P11, P12, P13, P14 and P15 in Table 4 shows that, the value of NCCM is in increasing order and the value of M is in decreasing order. With the increase in DIT from 1 to 4 the complexity increases. The value of M=4.20 with DIT=1 is much more than M=3.0 with DIT=4. But with NCCM, the value increases from

0.70 to 1.53 with the increase of DIT from 1 to 4. The graph in Fig. 6 shows negative slope with M & C and which is inverse to DIT and NCCM having positive slope.

TABLE IV. COMPARISON OF NCCM VALUE OF PROGRAM P8 TO P15

Program	Number of Nested Classes	DIT	NCCM	M	C
P8	4	1	0.70	4.20	3.00
P9	4	2	0.92	3.33	2.67
P10	4	2	0.92	3.75	2.83
P11	4	2	1.03	3.75	2.50
P12	4	3	1.17	3.33	2.58
P13	4	3	1.25	3.33	2.42
P14	4	3	1.28	3.33	2.33
P15	4	4	1.48	3.00	2.28

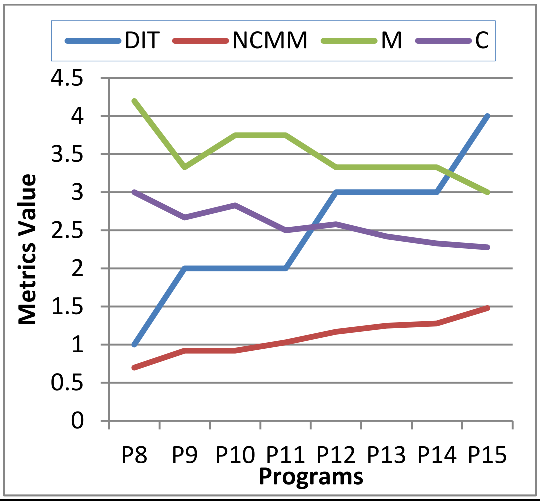

Figure 6. Metrics value of program P8 to P15.

V. CONCLUSIONS

Object oriented metrics help the developer in the object oriented software development. The various complexity metrics proposed by different researchers from time to time mainly depict the use of classes, inheritance, coupling, cohesion, and polymorphism factors in their research. Here, we have used nested classes or inner classes, which enhance encapsulation, motivate the developer to use them frequently. The proposed complexity metric was compared with existing metrics using different set of programs. It is quite interesting that in each case, the proposed complexity metric provides better results than the existing ones. The proposed metric in graphical representation shows positive slope with DIT whereas the other existing metrics show negative slope. This metric may be improved or some new metrics may be designed for nested classes in future by using some other aspects of object oriented software development.

REFERENCES

[1] S. R. Chidamber and C. F. Kemerer, “Towards a metric suite for object-oriented design,” in *Proc. the Conference on Object-Oriented Programming Systems, Languages and Applications*, ACM Press: NY, 1991, pp. 197-211.

[2] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994.

[3] S. R. Chidamber and C. F. Kemerer, “Managerial use of metrics for object-oriented software: An exploratory analysis,” *IEEE Transactions on Software Engineering*, vol. 24, no. 8, pp. 629-639, 1998.

[4] L. C. Braind and S. Morasoa, “Defining and validating measures for object-based high level design,” *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 722-743, 1999.

[5] R. Harrison, S. J. Counsell, and R. V. Nithi, “An evaluation of MOOD set of object oriented software metrics,” *IEEE Trans. Software Engineering*, vol. SE-24, no. 6, pp. 491-496,1998.

[6] R. S. Chhillar and P. Kajla, “Metrics to study constructor in class hierarchy,” in *Proc. National Conference on Advanced Computing Technologies-2013*, March 2013, vol. 2, pp. 923-926.

[7] W. Li and S. Henry, “Object-oriented metrics that predict maintainability,” *Journal of Systems and Software*, vol. 23, no. 2, pp. 111-122, 1993.

[8] M. H. Tang, M. H. Kao, and M. H. Chen, “An empirical study on object-oriented metrics,” in *Proc. 23rd Annual International Computer Software and Application Conference, IEEE Computer Society*, 1999, pp. 242-249.

[9] R. S. Pressman, *Software Engineering: A Practitioner’s Approach*, 6th ed., Singapore: McGraw-Hill, 2005, ch. 15, pp. 480-486.

[10] F. B. Abreu, “The MOOD metrics set,” presented at *the 9th European Conference on Object-Oriented Programming, Workshop on Metrics*, Aarhus, Denmark, 1995.

[11] K. Morris, “Metrics for object oriented software development,” Master’s thesis, M.I.T., Sloan School of Management, Cambridge, MA, 1998.

[12] K. K. Aggarwal, Y. Singh, and R. Malhotra, “Empirical study of object-oriented metrics,” *Journal of Object Technology*, vol. 5, no. 8, pp. 149-173, 2006.

[13] U. Chhillar and S. Bhasin, “Establishing relationship between complexity and faults for object-oriented software systems,” *International Journal of Computer Science Issues*, vol. 8, no. 5, pp. 437-442, 2011.

[14] R. Singh and P. S. Grover, “A new program weighted complexity metric,” in *Proc. International Conference on Software Engg.*, 1997, pp. 33-39.

[15] S. Mishra, “An object oriented complexity metric based on cognitive weights,” presented at 6th IEEE International Conference on Cognitive Informatics, California, USA, August 6 – 8, 2007.

[16] R. S. Chhillar, P. Ahlawat, and U. Chhillar, “Measuring complexity of component based system,” in *Proc. 2nd International Conference on Information Communication and Management*, vol. 55, 2012, pp. 19-27.

[17] N. S. Gill, *Software Engineering: Software Reliability, Testing and Quality Assurance*, New Delhi: Khanna Book Publishing Co., 2007, ch. 12, pp. 341-346.

[18] S. H. Tee, “Developing a maintainability metric for inner classes,” *Asian Journal of Information Technology*, vol. 9, no. 2, pp. 98-100, 2010,

[19] S. H. Tee, R. Atan, and A. Ghani, “Developing a complexity metric for inner classes,” *Journal of Theoretical and Applied Information Technology*, vol. 12, no. 2, pp. 77-83, 2010.

Dr. Rajender Singh Chhillar is working as Professor and Head, Department of Computer Science and Applications, Maharshi Dayanand University (MDU), Rohtak, Haryana, India. He acted as Director, University Institute of Engineering and Technology (UIET), M. D. University, Rohtak from April, 2006 to August 2007 and remained Head, Department of Computer Science and Applications, M. D. University, Rohtak earlier also from March 2003 to March 2006. He also worked as Director Computer Centre, MDU from 2003 to 2010. He was member, monitoring committee of campus wide Networking, M. D. University, Rohtak. He obtained his Ph.D. in Computer Science from Maharshi Dayanand University, Rohtak and Master’s Degree from Kurukshetra University, Kurukshetra. His researches include Software Engineering, Software Testing, Computer Network Security, Software Metrics, Component and Aspect based

Metrics, Data Warehousing and Data Mining, Information and Network security and IT Management. He has published more than 150 publications in International and National journals/ conferences. Professor Chhillar has also authored two books – Software Engineering: Metrics, Testing and Faults, Excel Books House, New Delhi; and Application of Information Technology to Business, Ramesh Books House, Jaipur. He is senior member of various National and International academic bodies/associations and reviewer of various International journals.

Mr. Parveen Kajla is a Research Scholar in the Department of Computer Science and Applications, Maharshi Dayanand University (MDU), Rohtak, Haryana, India. He is coordinator of PG Courses at Vaish Mahila Mahavidyalya, Rohtak and Senior Lecturer in Department of Computer Science and Applications, Vaish Mahila Mahavidyalya, Rohtak. He obtained his Master's Degree in Computer Science from Maharshi Dayanand University, Rohtak and M. Phil. (Computer Science) from Chaudhary Devi Lal University (CDLU), Sirsa. His research interest includes Software Engineering focusing on Object oriented and component based metrics.

Dr. Usha Chhillar is working as Head, Department of Computer Science, A.I.J.H.M. PG College, Rohtak, Haryana, India. She obtained her Ph.D. Degree in Computer Science from Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, Haryana, India. She pursued her Master Degree in Computer Science from Maharshi Dayanand University (MDU), Rohtak and M. Phil. (Computer Science) from Ch.Devi Lal University (CDLU), Sirsa. She has total more than thirteen years teaching experience. Her research interests include Software Engineering, Object-Oriented and Component-based Software Metrics.