IDENTIFYING THE RELATIONSHIP BETWEEN ORIGINAL AND REFINED POP METRIC FOR OO SOFTWARE

[#1]Vijay Yadav , [#2]Dr. Vibhash Yadav ,[#3]Prof. Raghuraj Singh
[#1]Assistant Professor (CSE), Kanpur Institute of Technology, Kanpur, UP India.
[#2]Assistant Professor (CSE), Kanpur Institute of Technology, Kanpur, UP India.
[#3]Assistant Professor (CSE), KIT, Sultanpur, UP India.
[#1]vijayyadavuiet@gmail.com,[#3] raghurajsingh@rediffmail.com

ABSTRACT

Most of the researchers have worked for size and effort evaluation but still the problem has not been fully resolved. Many of the existing evaluation techniques work specifically for specific development environment. PRICE systems has developed the predictive object point (POP) metric for predicting effort required for developing an object oriented software system and is based on the counting scheme of function point (FP) method. Though it was an interesting theoretical development, but due to absence of an easy to use support tool and too much complicated formulations, it could not gain sufficient recognition from practitioners to be use on a regular basis. In this paper, it is tried to show the relationship between the original POP and the refined POP formulations for POP calculation. This relationship helps in simplifying the POP count formula and preliminary results of its application in an industrial environment are presented and discussed here for validation of the suggested simplification in measurement of POP metric.

Index Terms: Software Measurement, Object Orientation, Functional size measurement, Software Metrics, Predictive Object Point, and Automation.

1. INTRODUCTION TO POP METRIC

POP was introduced by Mickiewicz in 1998. PRICE systems [8] has developed the POP metric for predicting effort required for developing an object oriented software system. POPs are suitable metrics for estimating the size and subsequently the effort required for development of object oriented software [10].It was designed specifically from results on measurement of the object-oriented properties for Object oriented software systems. It fulfilled almost all the criteria of OO concepts and was based on the counting scheme of function point (FP) method as used in function/procedure oriented software

development environment [1]. POPs are intended as an improvement over FPs by drawing on well-known metrics associated with an object oriented system [11].

2. CACULATION PROCSS OF POP METRIC

The following metrics are necessary for object-oriented systems in POP Count: Number of top level classes (TLC), Average number of weighted methods per class (WMC),Average depth of inheritance tree (DIT), and Average number of children per base class (NOC). WMC, DIT, and NOC are taken from the MOOSE metrics suite [3][5].

These metrics are then used to form the equation (1), giving the number of POPs for a system [8].

$$f1(TLC, NOC, DIT) = TLC*(1+ ((1+NOC)*DIT)^{1.01}+(|NOC-DIT|)^{.01})$$

$$f2(NOC,DIT)=1.0 \quad (1)$$

$$POPs(WMC, NOC, DIT, TLC)= \frac{WMC* f1(TLC, NOC, DIT) * f2(NOC, DIT)}{7.8}$$

Where, f1 attempts to size the overall system, and f2 applies the effects of reuse through inheritance.

3. SUGGESTED RELATIONSHIP BETWEEN ORIGINAL POPAND REFINED POP

An easy to use automation tool APA (Automated POP Analyzer) is built for counting POPs by splitting the whole Java Project into files and calculating POP on the basis of its individual java file. In the True OO environment as in java projects, the level of reusability through Inheritance is always considered to be high and hence function of NOC and DIT can be considered as 1.0 [2]. Thus the correction factor f_2 taken by Mickiewicz [8] can be omitted while estimating Java projects. However this may not be true for other environments.Thus the factor $|NOC-DIT|^{.01}$ may be omitted and f2 may be neglected while calculating POP Count values for Java Projects. The POP Count formula may be reduced to the equation (2).

$$.01 f1(TLC, NOC, DIT) = TLC * (1 + ((1 + NOC) * DIT))^{1.01} \quad (2)$$

$$POPs(WMC, NOC, DIT, TLC)= \frac{WMC * f1(TLC, NOC, DIT)}{7.8}$$

Further the simplification in POP count formula has been suggested and validated by the introduction of the new OO metric AWC (Average Weighted Method Count) which can be used to replace the WMC (Weighted Method Count) metric as shown in equation (3).

$$WMC=AMC \times 10.478 \quad (3)$$

which involves very rigorous method of calculation [1]. In this paper it has been tried to identify the relationship between Original POP count which is calculated by using equation (1) and the Refined POP count which is calculated from equation (2). This can be used for java projects. However this may not be true for other environments[1].

4. POP COUNT ESTIMATION PROCES

The following process was followed for calculation of POP Count:

A. Step 1: The first step was to obtain the Source Lines of Code (SLOC) metric for projects through APA tool [4] based on CCCC, an object oriented metric gathering tool [7].

B. Step 2: Using the generated DIT metrics for each class it was possible to calculate the average DIT (one of the metrics required for POPs). Similarly the generated NOC metrics for each class were averaged to obtain the average NOC.

Average NOC = (Sum of Base Class NOCs) / (Number of Base Classes giving +ve NOC count.)

Average DIT = (Sum of Classes having DITs) / (Sum of the rows of NOC and DIT giving +ve count).

C. Step 3: Average Method count (AMC) is calculated by dividing the method count by the class count [4].

D. Step 4: The TLC metric for each java file and for overall project was then calculated. This includes the base classes (with no parents) and the class which is at level 0. This metric is a count of the classes that are roots in the class diagram, from which all other classes are derived [8].

E. Step 5: Finally WMC is calculated as suggested by Minkiewicz [8]. As in order to determine the average number of methods in each type, weightings should be applied against this as per the following calculations [9]:

Average Constructor/Destructor Method Count = 20% (Average Methods per Class)

Average Selector Method Count = 30% (Average Methods per Class)

Average Modifier Method Count = 45% (Average Methods per Class).

Average Iterator Method Count = 5% (Average Methods per Class).

Now, each method type was divided into three categories of complexity using weightings. This spread of method types

arose from a manual investigation of source code by Minkiewicz [8].

Low Complexity Method Count = 22% of Average Method Count

Average Complexity Method Count = 45% of Average

Method Count High Complexity Method Count = 33% of Average Method Count

For each java file all twelve calculations were performed and their sum gives the value of WMC [9]. The same method is used for the calculation of WMC for the overall project.

5. DESCRIPTION OF EMPIRICAL STUDY

The proposed relationship between Original POP and Refined POP count for Java Projects can be validated, under this study, 62 projects including 2 projects from research work of T. R Judge and A. Williams [6] as shown in Table 1 have been considered.

Project No.	Project Name	Original POP	Refined POP	Ratio Ori/Ref
1.	ATM_Banking_Sys	119.2381	82.9576	1.44
2.*	Band_Width_Esti	302.831	190.4707	1.59
3.*	Face_Detection_Syst	50.0243	31.2973	1.59
4.	Face_Id	173.9161	112.7132	1.54
5.*	Face_book_Like_Chat	156.554	99.0489	1.58
6.	Flight_Reserv_System	86.4455	57.7633	1.50
7.	JaimBot_Ver_1.2	561.852	379.1025	1.48
8.	JaimBot_Ver_1.2.1	597.89	403.0064	1.48
9.	JaimBot_Ver_1.3	661.55	444.0895	1.49
10.	JaimBot_Ver_1.4	1000.88	674.1781	1.48
11.	JaimLib_Ver_0.4	903.65	581.582	1.55
12.	JaimLib_Ver_0.5	910.797	586.0526	1.55
13.*	Library_Mgt_Sys	386.61	236.6697	1.63
14.	Library_Sys	411.611	262.5484	1.57
15.*	Medical_Diag_Sys	187.27	117.7235	1.59
16.*	Mobile_Pay_Service	924.84	585.1817	1.58
17.	Online_Address_Book	160.829	107.4667	1.49
18.*	Payroll	312.3165	191.1587	1.63
19.*	Civil_Game_Java	510.9983	318.3434	1.60
20.	Remote_Adm_Sys	41.54	26.8449	1.55
21.	PhysicsMata_ver_0.1.2	44.2279	29.5533	1.50
22.*	PhysicsMata_ver_0.3.0	128.64	79.1175	1.63
23.*	PhysicsMata_ver_0.3.1	128.64	79.1175	1.63
24.*	PhysicsMata_ver_0.4.1	208.712	133.1823	1.57
25.	PhysicsMata_ver_0.5.0	249.2617	159.7175	1.56
26.	PhysicsMata_ver_0.5.1	253.2828	162.4041	1.56
27.*	PhysicsMata_ver_0.5.2	698.737	429.7375	1.63
28.	PhysicsMata_ver_0..6.0	24.124	16.12	1.50
29.	PhysicsMata_ver_0..6.1	24.124	16.12	1.50
30.	PhysicsMata_ver_0.8.0	24.124	16.12	1.50
31.	PhysicsMata_ver_0.8.1	24.124	16.12	1.50
32.	PhysicsMata_ver_1.2.0	56.29	37.6133	1.50
33.	PhysicsMata_ver_1.2.1	56.29	37.6133	1.50
34.	JavaGeom_ver_0.3.0	2242.0278	1452.1516	1.54
35.	JavaGeom_ver_ 0.3.2	2417.585	1566.2636	1.54

36.	JavaGeom_ver_0.5.0	2543.312	1635.5748	1.55
37.	JavaGeom_ver_ 0.5.2	3191.8688	2036.503	1.57
38.	JavaGeom_ver_0.5.1	2539.5093	1631.7403	1.57
39.	JavaGeom_ver_ 0.6.0	2845.0487	1833.1131	1.55
40.	JavaGeom_ver_0.6.2	2876.133	1864.2768	1.54
41.	JavaGeom_ver_ 0.6.3	3439.5835	2205.3822	1.56
42.*	JavaGeom_ver_0.7.0	4521.7321	2793.9842	1.62
43.*	JavaGeom_ver_ 0.7.1	4580.0669	2829.5591	1.62
44.*	JavaGeom_ver_0.8.0	3952.02153	2432.8656	1.62
45.*	JavaGeom_ver_0.8.1	4357.4644	2677.6644	1.63
46.	Lwjgl_alpha_0.3	1256.5475	824.808	1.52
47.	Lwjgl_0.4	1705.7530	1125.2742	1.52
48.	Lwjgl_0.5	1380.0746	907.6542	1.52
49.	Lwjgl_0.6	2222.9296	1434.6609	1.55
50.*	Lwjgl_0.92	3978.5015	2464.6488	1.61
51.*	Lwjgl_0.93	4001.2856	2479.8732	1.61
52.*	Lwjgl_0.95	5526.3933	3452.9299	1.60
53.*	Lwjgl_0.96-2	6914.9014	4348.3864	1.59
54.*	Lwjgl_0.97-1	6941.0338	4365.1641	1.59
55.*	Lwjgl_0.98-1	6915.5922	4355.0044	1.59
56.*	Lwjgl_0.99	7140.7526	4505.4578	1.58
57.*	Lwjgl_1.0	7805.3684	4926.3582	1.58
58.*	Lwjgl_1.0beta4	7777.6232	4910.0088	1.58
59.*	Lwjgl_1.0 -rc1	7792.5040	4918.8552	1.58
60.*	Lwjgl_1.1	751.8859	889.5221	1.59

Table 1: Projects analyzed to study Relationship between Original and Refined POP

The Project marked with * in corresponding S. No. give approx value of 1.6 to the ratio of Original and Refined POP.

Execution of the project javaGeom-0.5.2-src shown in Fig.1.1 and Fig.1.2 on APA(Automated POP analyzer) tool, this tool is developed and automated by us for analyzing the POP metric, The values of Original POP and Refined POP are highlighted as under the circled value in the snapshot. The value of the Original POP for this project is found to be 3191.8688 and the value of Refined POP is 2036.503 as shown in Fig 1.1 and Fig 1.2

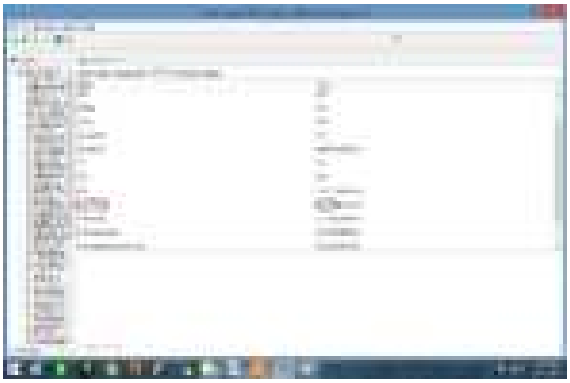

Fig.1.1 Original POP Values

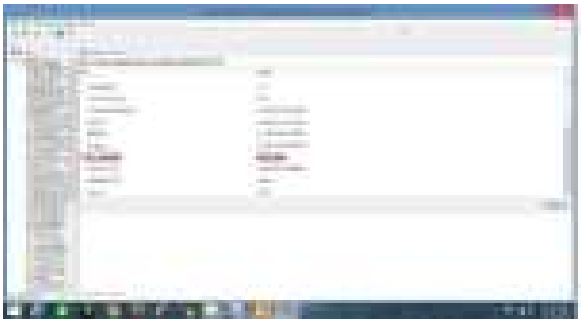

Fig.1.2 Refined POP values

Project Attributes	Project Alpha	Project Beta
Source Lines of Code (SLOC)	38854	20570
Total Number of Classes	404	147
Total Number of Methods	2412	833
Average of the Methods per Class	5.971	5.667
Average Depth of Inheritance	0.941	0.701
Average Number of Children	3.700	2.688
Top Level Classes	201	73
Constructors/Destructors (20%)	1.194	1.133
Selectors (30%)	1.791	1.700
Modifiers (45%)	2.687	2.550
Iterators (5%)	0.299	0.283
WMC	62.564	59.379
Number of POPs Original	10478	2566

Table 2: Summary of Project Metrics [6]

The proposed relationship between Original POP and Refined POP formulations can further be checked in reference to the projects taken by T. R Judge and A. Williams [6]. In their research work using projects Alpha and Beta, they proved POP metric as better indicator of software size in comparison to FP metric as shown in Table

Project Attributes	Project Alpha	Project Beta
Number of refined POPs	8849.422	2006.1515

Table 3:Refined POP Count for Projects

Project Name	Original POP	Refined POP	Ratio
Alpha	10478	8849.422	1.2
Beta	2566	2006.1515	1.3

Table:4 Ratio of the Original POP and Refined POP

On analyzing the above projects, it was found that the ratio of the Original POP and Refined POP most of the times comes out to be approx 1.6 as seen from the Table 1. Thus there can be a constant k whose value may be approx 1.6 to give the relationship between Original POP and Refined POP as in equation (3).

The result from Table 5 gives the value of the constant k to be 1.2 and 1.3 for projects alpha and beta respectively. So from this it can be assumed that the value of the constant k will lies between 1.2 to 1.6.

Hence this further validates the proposed relationship between Original POP and Refined POP count.

6. CONCLUSIONS

One of the factors that affected the adoption of POP methods in practice was the lack of support tools to help estimators in their tasks. Another problem was the complicated formulation of POP count. Here, in this paper, the relationship between the Original POP count and the Refined POP count formula has been proposed through which POP metrics calculations have been simplified. The projects taken for empirical study from research work of T. R Judge and A. Williams [6], are also considered for proposing the value of the constant k, however the data to be studied may include additional java projects. This will further ensure the validity for this relationship for Java Projects and hence accuracy of the measurement.

REFERENCES

1. Er. Vijay Yadav., Dr. Vibhash Yadav., Prof. Raghuraj Singh,(2016),. "Introducing New OO Metric For Simplification In Predictive Object Points (POP) Estimation Process in OO Environment ". International Journal of Engineering Sciences & Research Technology, ISSN : 2277-9655 (I2OR), Publication Impact factor: 3.785., Volume-5(1), Issue-January,2016.
2. Shubha Jain, Vijay Yadav, Raghuraj Singh., "Assessment of Predictive Object Points (POP) Values for Java Projects". International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970) Volume-3 Number-4 Issue-13 December-2013 Impact Factor 1.863. Available At:www.theaccents.org/ijacr/currentissue.html. (Published).
3. Shubha Jain, Vijay Yadav, Raghuraj Singh., "A Simplified Formulation of Predictive Object Points(POP) Sizing Metric For OO Measurement". IACC 2014 4th IEEE International Advance Computing Conference, Part Number CFP1439F-CDR ISBN 978-1-4799-2571-1, Feb 21st-22nd 2014, IEEE Computer Society, Gurgaon, India.(Published in IEEE Xplorer)..
4. Shubha Jain, Vijay Yadav and Prof. Raghuraj Singh, (2013). "OO Estimation Through Automation of

Predictive Objective Points Sizing Metric"., International Journal Of Computer Engineering and Technology (IJCET) Volume 4, Issue 3, May- June (2013), pp. 410-418Shyam R. Chidamber and Chris F. Kemerer, "A Metrics Suite for Object Oriented Design". IEEE transactions On Software Engineering, Vol. 20, No. 6, pp. 476-493, June 1994.

5. T. R Judge, A. Williams, (2001), "OO Estimation – an Investigation of the Predictive Object Points (POP) Sizing Metric in an Industrial Setting". Parallax Solutions Ltd, Coventry, UK.,
6. CCCC Metric Tool by Tim Littlefair. http://www.fste.ac.cowan.edu.au/~tlittlef/
7. Arlene F. Minkiewicz,(1997), "Object-Oriented Metrics" Software Development. Wiley Computer Publishing, pp. 43-50,1997 at: http://www.sdmagazine.com
8. Shubha Jain, Vijay Yadav and Prof. Raghuraj Singh, (2013). "OO Estimation Through Automation of Predictive Objective PointsSizing Metric"., International Journal Of Computer Engineering
9. Shubha Jain, Vijay Singh., " An Yadav, Raghuraj Approach
10. International Conference on "Computing for Sustainable Global Development. Paper ID: 57, ISSN 0973-7529 and ISBN 978-93-80544-10-6 serials, IEEE Conference ID 32558, IEEE sponsors: Delhi Section, Other sponsor: Bharati Vidyapeeth Institute of Computer Applications and Management, New Delhi, India. Available At: www.bvicam.ac.in/indiacom/(Published)
11. Haugh M. E. W Olsen and Bergman. L. "Software Process Improvement: Metrics Measurement and Process Modeling", Vol. 4, New York, Springer, pp. 159-170, 2001.