A Simplified Formulation of Predictive Object Points (POP) Sizing Metric For OO Measurement

Shubha Jain, Vijay Yadav
Department of Computer Science and Engineering
Kanpur Institute of Technology,
Kanpur, India.
shubhj@rediffmail.com, vijayyadavuiet@gmail.com

Prof. Raghuraj Singh
Computer Science and Engineering Department
Harcourt Butler Technological Institute,
Kanpur, India.
raghurajsingh@rediffmail.com

Abstract— **Different effort estimation techniques exist for sizing software systems but none is directly applicable to object-oriented software. Although many researchers worked for size and thus effort estimation but still the problem not resolved fully. Different existing estimation techniques works 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 could not gain sufficient recognition from practitioners to be used on a regular basis due to lack of an easy to use support tool and too much complicated formulations. In this paper we tried to simplify the POP calculation. The POP count formula suggested by PRICE system has been modified for estimating the effort. An easy to use support tool to automate the counting method has been prepared. The refined POP count formula and preliminary results of its application in an industrial environment are presented and discussed here for validation of the suggested modification or simplification in formula.**

Keywords—Software Measurement, Object Orientation, Functional size measurement, Software Metrics, Predictive Object Point, Automation

I. INTRODUCTION TO MEASUREMENT

Measurement can be considered as mapping from the empirical world to the formal, relational world. It is the number or symbol assigned to an entity by this mapping in order to characterize an attribute. Measurements are associated with improvements. The paradigm turned out to be "if you can't measure, you can't improve" [1]. In the scientific fields, including engineering, measurement is considered as one of a number of analytical tools and is based on a large body of knowledge built up over centuries [11] [12].

Cost and effort estimation is an important aspect of the management of software development projects. An accurate estimation is difficult: an average error of 100% may be considered "good" and an average error of 32% "outstanding" [18]. Most methods for estimating effort require an estimation of the size of the software. Once a size estimate is available, models can be used that relate size to effort [17].The introduction to software metrics came with a job move from a normal to commercial software engineering environment. Moving from measurements to metrics is like moving from observation to understanding [13]. The metrics are guidelines and not rules. They give an indication of the progress that a project has made and the quality of the design [14].

Metrics are also indicators and help in taking data driven decisions in time. By the implementation of Object Oriented Paradigm the researchers modified and validated the conventional metrics theoretically or empirically. Sizing and complexity metrics were the most impressive contributions for effort and cost estimation in project planning [15] [16].

II. POP METRIC OVERVIEW

POP was introduced by Mickiewicz in 1998. PRICE systems [2] has developed the predictive object point (POP) metric for predicting effort required for developing an object oriented software system. It was designed specifically for Object oriented software and result from measurement of the object-oriented properties of the system. It fulfilled almost all the criteria of OO concepts and was based on the counting scheme of function point (FP) method. POPs are intended as an improvement over FPs, which were originally intended for use within procedural systems, by drawing on well-known metrics associated with an object oriented system [3]. POPs are a metric suitable for estimating the size, subsequently effort of object oriented software, based on the behaviors that each class is delivering to the system along with top level inputs describing the structure of a system.

A. Abstract Model

POPs combine the following metrics presented in the literature to measure object-oriented systems: 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 of Chidamber and Kemerer [7].

B. Measurement Process

The above mentioned metrics are then gathered to form the equation (1), giving the number of POPs for a system [2].

978-1-4799-2572-8/14/$31.00 ©2014 IEEE

$$f1(TLC, NOC, DIT) = TLC * (1 + ((1 + NOC) * DIT)^{1.01} + (| NOC - DIT |)^{.01})$$
$$f2(NOC, DIT) = 1.0 \quad (1)$$
$$POPs(WMC, NOC, DIT, TLC) = \frac{WMC * f1(TLC, NOC, DIT)}{7.8} * f2(NOC, DIT)$$

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

C. Proposed Refinement in POP Count Calculation

Estimation through POP count could not gain sufficient recognition from practitioners to be used on a regular basis due to the lack of an easy to use support tool and too much complicated formulations. POPs use in real life was impractical as it required entirely too much detail for top down estimation [4].

Predictive Object points have been developed in the context of companies (e.g. Price Systems). But no details about their actual usage in these companies are publicly available [4]. This research was conducted nearly 20 years ago and unfortunately has been discontinued.

An easy to use automation tool APA (Automated POP Analyzer) is used for counting POPs and results have been validated by introducing modifications in APA tool [4].

A refined formula for POP Count calculation has been proposed in equation (2).

$$f1(TLC, NOC, DIT) = TLC * (1 + ((1 + NOC) * DIT)^{1.01})$$
$$f2(NOC, DIT) = 1.0 \quad (2)$$
$$POPs(WMC, NOC, DIT, TLC) = \frac{WMC * f1(TLC, NOC, DIT)}{7.8} * f2(NOC, DIT)$$

In this paper the simplification in POP count formula has been suggested and validated.

III. POP METRIC AUTOMATION

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 [5].

B. Step 2

Using the generated DIT metrics for each class it was possible to calculate the average DIT (a metric required for POP). 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 [2].

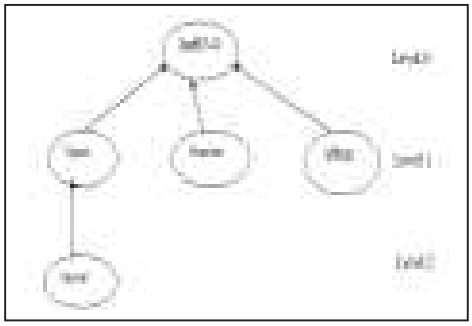

Fig .1.1 Class Diagram of EducationSystem.cpp

Average DIT= ((1*0) + (3*1) + (1*2))/5=1
Average NOC= ((1*3) + (1*1)/2=2
TLC=1.

Execution of the project Education System shown in Fig.1.1 on APA tool, gives the same result as shown in Fig.1.2. The results are highlighted as the circled value in the snapshot.

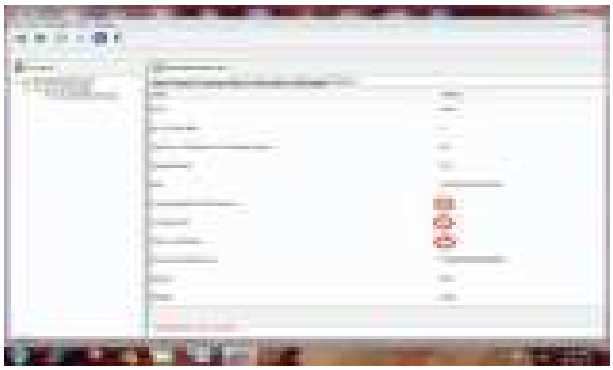

Fig .1.2 Sample Avg DIT, Avg NOC and TLC Values through APA Tool

E. Step 5

Finally WMC is calculated as suggested by Minkiewicz [2]. As in order to determine the average number of methods in each type, weightings should be applied against this as per the following calculations [1]:

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 [1].

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 [4]. The same method is used for the calculation of WMC for the overall project.

IV. DESCRIPTION OF EMPIRICAL STUDY

64 projects including 2 projects from research work of T. R Judge and A. Williams [6] for POP validation have been considered for this study shown in Table I.

TABLE I. PROJECT ANALYZED TO STUDY REFINEMENT IN POP CALCULATION

Project No.	Project Name	No. of Java Files	SLOC	TLC
1.	ATM_Banking_Sys	12	1186	6
2.	Band_Width_Esti	8	2584	11
3.	Face_Detection_Syst	3	600	2
4.	Face_Id	11	2461	21
5.	File_Compression	33	2319	22
6.	Face_book_Like_Chat	6	695	9
7.	Flight_Reserv_System	6	719	3
8.	JaimBot_Ver_1.2	32	2314	20
9.	JaimBot_Ver_1.2.1	23	2663	21
10.	JaimBot_Ver_1.3	24	2880	23
11.	JaimBot_Ver_1.4	30	4413	33
12.	JaimLib_Ver_0.4	45	1505	44
13.	JaimLib_Ver_0.5	45	1539	44
14.	Library_Mgt_Sys	17	2859	43
15.	Library_Sys	30	4552	43
16.	Medical_Diag_Sys	9	2013	10
17.	Mobile_Pay_Service	32	2887	47
18.	Online_Address_Book	12	614	12
19.	Payroll	9	1834	13
20.	Civil_Game_Java	17	2659	18
21.	Remote_Adm_Sys	3	383	12
22.	PhysicsMata_ver_0.1.2	4	169	2
23.	PhysicsMata_ver_0.1.3	6	319	4
24.	PhysicsMata_ver_0.3	8	239	7
25.	PhysicsMata_ver_0.3.1	8	237	7
26.	PhysicsMata_ver_0.4.1	31	492	32
27.	PhysicsMata_ver_0.5.0	11	398	9
28.	PhysicsMata_ver_0.5.1	12	416	9
29.	PhysicsMata_ver_0.5.2	34	1168	29
30.	PhysicsMata_ver_0..6.0	5	224	6
31.	PhysicsMata_ver_0..6.1	5	227	6
32.	PhysicsMata_ver_0.8.0	12	227	7
33.	PhysicsMata_ver_0.8.1	10	230	7
34.	PhysicsMata_ver_1.2.0	19	418	15
35.	PhysicsMata_ver_1.2.1	19	419	15
36.	JavaGeom_ver_0.3.0	53	2687	57
37.	JavaGeom_ver_ 0.3.2	55	3179	60
38.	JavaGeom_ver_0.5.0	67	2888	80
39.	JavaGeom_ver_ 0.5.2	78	3311	93
40.	JavaGeom_ver_0.5.1	72	2868	80
41.	JavaGeom_ver_ 0.6.0	74	3190	91
42.	JavaGeom_ver_0.6.2	78	3725	85
43.	JavaGeom_ver_ 0.6.3	84	3919	85
44.	JavaGeom_ver_0.7.0	88	4315	114
45.	JavaGeom_ver_ 0.7.1	94	4359	117
46.	JavaGeom_ver_0.8.0	104	3869	134
47.	JavaGeom_ver_0.8.1	107	4053	134
48.	Lwjgl_alpha_0.3	41	6463	22
49.	Lwjgl_0.4	61	9397	39
50.	Lwjgl_0.5	47	7578	25
51.	Lwjgl_0.6	67	10363	48
52.	Lwjgl_0.92	266	18262	96
53.	Lwjgl_0.93	273	19366	96
54.	Lwjgl_0.95	291	19555	116
55.	Lwjgl_0.96-2	443	23580	163
56.	Lwjgl_0.97-1	443	23682	163
57.	Lwjgl_0.98-1	450	24640	165
58.	Lwjgl_0.99	453	25744	164
59.	Lwjgl_1.0	503	30097	179
60.	Lwjgl_1.0beta4	511	29558	179
61.	Lwjgl_1.0 -rc1	529	30542	179
62.	Lwjgl_1.1	544	32219	184

Table II shows the number of methods, classes, AMC, WMC metrics as well as value of a factor $|NOC\text{-}DIT|^{.01}$ needed for POP Count calculations for all chosen projects.

TABLE II. METRIC VALUES FOR CHOSEN PROJECTS

Project No.	Methods	Classes with+ve method count	AMC	WMC	$\|NOC\text{-}DIT\|^{.01}$
1.	44	12	3.33	31.046	0.9958
2.	80	12	6.667	62.868	0.9955
3.	30	3	10.0	104.78	0.9960
4.	67	29	2.3103	25.6235	0.9956
5.	188	29	6.483	72.7415	0.9937
6.	42	9	4.667	52.39	0.9943
7.	53	7	7.5714	86.4435	0.9931
8.	191	33	5.7879	70.2026	0.9923
9.	208	35	5.9429	71.39	0.9923
10.	227	37	6.1351	72.53	0.9949
11.	352	50	7.04	81.54	0.9949
12.	207	45	4.6	48.199	0.9939
13.	210	45	4.667	48.897	0.9939
14.	89	31	2.871	31.74	0.9963
15.	134	52	2.5769	28.29	0.9947
16.	59	13	4.5385	44.53	0.9942
17.	152	32	4.75	49.771	0.9946
18.	40	12	3.33	34.9267	0.9931
19.	57	13	4.385	48.894	0.9963
20.	124	19	6.526	70.881	0.9944
21.	160	24	6.667	61.667	0.9946
22.	20	5	4.0	48.671	0.9931

23.	38	7	5.4286	61.9948	0.9931
24.	20	4	5.0	52.39	0.9940
25.	20	4	5.0	52.39	0.9940
26.	54	15	3.6	38.544	0.9944
27.	56	10	5.6	58.677	0.9939
28.	59	11	5.3636	56.20	0.9938
29.	144	27	5.33	55.883	0.9941
30.	36	4	9	94.302	0.9950
31.	37	4	9.25	96.9215	0.9949
32.	42	7	6.0	62.868	0.9945
33.	33	6	5.5	57.629	0.9945
34.	33	15	2.2	23.052	0.9937
35.	33	15	2.2	23.052	0.9946
36.	495	50	9.9	103.711	0.9936
37.	531	52	10.21	107.039	0.9936
38.	533	58	9.1897	95.9564	0.9939
39.	616	62	9.9355	103.7494	0.9942
40.	530	56	9.4364	98.874	0.9940
41.	592	62	9.5484	99.7128	0.9942
42.	689	63	10.9365	116.35	0.9941
43.	690	59	11.6949	124.8152	0.9942
44.	738	69	10.6957	114.7123	0.9948
45.	745	70	10.6429	114.0759	0.9947
46.	654	76	8.6053	90.8585	0.9950
47.	699	78	8.6697	94.8283	0.9949
48.	548	54	10.148	126.046	0.9949
49.	679	75	9.0533	106.6204	0.9942
50.	599	61	9.8197	120.5448	0.9947
51.	748	81	9.2346	107.5448	0.9945
52.	2418	276	8.9542	93.8424	0.9939
53.	2543	284	8.9542	95.914	0.9939
54.	2864	298	9.611	101.9141	0.9940
55.	3247	381	8.5223	90.9796	0.9939
56.	3196	380	8.4105	89.7223	0.9939
57.	3338	450	8.4081	90.5409	0.9939
58.	3541	401	8.8304	95.4159	0.9939
59.	4059	463	8.7667	95.1577	0.9941
60.	4072	470	8.6638	93.9971	0.9941
61.	4118	484	8.5082	92.0353	0.9941
62.	4228	499	8.4729	90.7256	0.9941

On analyzing the above projects, it was found that the factor $|NOC\text{-}DIT|^{.01}$ always comes out to be nearly 1.0 for all projects. Thus the factor $|NOC\text{-}DIT|^{.01}$ may be considered as 1 and may be neglected in the original POP calculation.

Table IV shows the Analysis Report of POP Count calculated using original formula and Refined POP count calculated from the amended formula by neglecting the factor $|NOC\text{-}DIT|^{.01}$ in calculation of f_1 and thus in calculation of POP Count. Effort for comparison with Original POP and Refined POP has been calculated through the Constructive Cost Model (COCOMO) which was launched in 1981 by Barry Boehm [8]. It is often mentioned as COCOMO 81 to discern from its follow-up, this model assumes that the size of the project can be estimated in thousands of delivered source instructions (KLOC) and then uses a non-linear equation to determine the effort for projects[9][10]. The formula is given by equation (3).

$$\text{Effort} = a*(\text{KLOC})^{b}\ \text{PM} \qquad (3)$$

The parameters a and b are those that hold the interest. The idea is to define them based on the characteristics of project. These characteristics are then compared with historical data to yield the right numbers for these parameters. This model is really simple. Boehm's [1981] considered three modes of software development in this model: organic, semi-detached and embedded, these are discussed in Table III.

TABLE III. THE COMPARISON OF THREE COCOMO MODES [10]

Mode	a	b	Project size	Nature of Project
Organic	2.4	1.05	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. e.g., Payroll, Inventory projects etc.
Semidetached	3.0	1.12	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar projects. e.g., database systems
Embedded	3.6	1.20	Typically over 300 KLOC	Large projects, Real time systems, Complex Interfaces, Very little previous experience. e.g., ATMs systems.

The effort for each java file is calculated and averaged to get the effort of whole project, based on the characteristic of project being considered.

TABLE IV. ORIGINAL POP, REFINED POP AND EFFORT VALUES FOR CHOSEN PROJECTS

Project No.	Original POP	Refined POP	Effort Calculated
1	119.2381	82.9576	2.7146
2	302.831	190.4707	5.9779
3	50.0243	31.2973	1.34
4	173.9161	112.7132	5.541
5	620.573	112.7651	4.961
6	156.554	99.0489	1.531
7	86.4455	57.7633	1.5697
8	561.852	379.1025	5.11
9	597.89	403.0064	5.8489
10	661.55	444.0895	6.33
11	1000.88	674.1781	9.78
12	903.65	581.582	3.104
13	910.797	586.0526	3.18
14	386.61	236.6697	6.33
15	411.611	262.5484	10.02
16	187.27	117.7235	4.616
17	924.84	585.1817	6.185
18	160.829	107.4667	1.271
19	312.3165	191.1587	5.8266
20	510.9983	318.3434	6.0206
21	41.54	26.8449	0.838
22	44.2279	29.5533	0.3495
23	96.4973	87.3076	0.6666
24	128.64	79.1175	0.5041
25	128.64	79.1175	0.4996
26	208.712	133.1823	1.0085
27	249.2617	159.7175	0.83065
28	253.2828	162.4041	0.8678
29	698.737	429.7375	2.4548
30	24.124	16.12	0.4656
31	24.124	16.12	0.4722
32	24.124	16.12	0.570
33	24.124	16.12	0.472
34	56.29	37.6133	0.8539
35	56.29	37.6133	0.8561
36	2242.0278	1452.1516	5.697
37	2417.585	1566.2636	6.8035
38	2543.312	1635.5748	6.0815
39	3191.8688	2036.503	7.0163
40	2539.5093	1631.7403	6.06089

41	2845.0487	1833.1131	6.7515
42	2876.133	1864.2768	7.9378
43	3439.5835	2205.3822	8.4247
44	4521.7321	2793.9842	9.275
45	4580.0669	2829.5591	9.3723
46	3952.02153	2432.8656	8.271
47	4357.4644	2677.6644	8.6697
48	1256.5475	824.808	15.0518
49	1705.7530	1125.2742	21.4699
50	1380.0746	907.6542	17.4909
51	2222.9296	1434.6609	23.6046
52	3978.5015	2464.6488	39.5774
53	4001.2856	2479.8732	42.0379
54	5526.3933	3452.9299	42.3882
55	6914.9014	4348.3864	50.9707
56	6941.0338	4365.1641	51.2118
57	6915.5922	4355.0044	53.32014
58	7140.7526	4505.4578	55.8616
59	7805.3684	4926.3582	65.3002
60	7777.6232	4910.0088	64.0397
61	7792.5040	4918.8552	66.2093
62	7751.8859	4889.5221	69.9503

On Comparing the POP count for any two projects say P_1/P_2 reveals how much times the project P_1 is of project P_2 in terms of size.

This POP Count ratio has been evaluated for different combinations of projects using both the original POP Count value and Refined POP Count Values. Both ratios are compared with the Effort ratios for the same projects. Fig .1.3 shows the sample POP original and refined values.

Fig.1.3 POP original and refined values

The closer this ratio to that of the efforts ratio, the more accurate the POP technique is. This is because Effort and POP count are proportional, where the constant of proportionality is the POP productivity rate [6]. These results are summarized in Table V.

TABLE V. COMPUTED EFFORT, POP AND SLOC RATIOS FOR VARIOUS PROJECTS

S.No	Project Compared	SLOC Ratio	Original POP Ratio	Refined POP Ratio	Effort Ratio
1.	P1| P2	0.4589	0.3937	0.4355	0.4541
2.	P3| P4	0.2438	0.2876	0.2776	0.2481
3.	P4| P5	1.0621	0.2803	0.9995	1.1169
4.	P6| P7	0.9666	1.8110	1.1458	0.9753
5.	P8| P9	0.8689	0.9397	0.9405	0.8737
6.	P10| P9	1.0815	1.1065	1.1017	1.0823
7.	P11| P10	1.5322	1.513	1.518	1.5450
8. *	P13| P12	1.0226	1.008	1.008	1.0245
9.	P15| P14	1.592	1.0647	1.1093	1.5829
10.	P15| P16	2.2613	2.1979	2.2302	2.1707
11.	P17| P18	4.7019	5.7505	5.4452	4.8662
12.	P19| P21	4.788	7.5185	7.1209	6.9529
13.*	P23| P22	1.8875	2.1818	2.1818	1.9072
14. *	P24| P25	1.008	1.0	1.0	1.009
15.	P26| P24	2.059	1.6225	1.6825	2.001
16.	P28| P27	1.0452	1.0161	1.01682	1.0447
17. *	P29| P25	4.9283	5.4317	5.4316	4.9135
18. *	P31| P30	1.0134	1.0	1.0	1.0142
19. *	P32| P33	0.9869	1.0	1.0	1.2076
20. *	P35| P34	1.0023	1.0	1.0	1.0025
21.	P37| P36	1.183	1.0783	1.07858	1.1942
22.	P38| P36	1.0748	1.1343	1.04425	1.0675
23.	P39| P38	1.1465	1.2550	1.2451	1.1537
24.	P41| P40	1.1123	1.1203	1.1234	1.1139
25.	P42| P36	1.3863	1.2828	1.2838	1.3933
26.	P43| P36	1.4585	1.5341	1.5187	1.4789
27.	P43| P38	1.3569	1.3524	1.3484	1.3853
28.	P43| P40	1.3665	1.3544	1.3515	1.3900
29.	P43| P42	1.0521	1.1959	1.1829	1.0613
30.	P44| P39	1.3032	1.4166	1.3719	1.3219
31.	P45| P44	1.0102	1.0129	1.0127	1.0105
32.	P46| P40	1.3490	1.5562	1.4909	1.3647
33.	P46| P38	1.3397	1.5539	1.4875	1.3600
34.	P47| P46	1.0475	1.1026	1.1004	1.0482
35.	P49| P48	1.4539	1.3575	1.3643	1.4264
36.	P49| P50	1.2400	1.2359	1.2398	1.2275
37.	P50| P48	1.1725	1.09831	1.1004	1.16205
38.	P51| P49	1.1028	1.3032	1.2749	1.1099
39.	P56| P54	1.2110	1.2642	1.2559	1.2442
40.	P58| P55	1.0794	1.03266	1.03612	1.09596
41.	P59| P58	1.1691	1.0930	1.0934	1.1689
42.	P60| P58	1.1482	1.0892	1.0898	1.1464
43.*	P61| P59	1.0148	0.9984	0.9985	1.0139
44.*	P61| P60	1.0333	1.002	1.002	1.0339
45.	P62|P61	1.0549	0.9948	0.9940	1.0565

The Project ratios marked with * in corresponding S.No. give similar results with original as well as refined POP.

From the results it may be seen that on neglecting the factor $|NOC\text{-}DIT|^{.01}$ in original formula of POP calculation either give same or better results near to effort. Thus this validates the proposed refined POP Count formula.

The proposed refined POP formula can further be checked in reference with 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 VI.

TABLE VI. SUMMARY OF PROJECT METRICS [6]

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	10478	2566

They summarize the effort metrics that were originally estimated by the development team and the effort that was actually expended. These statistics are presented as ratios [6]:

(Actual Effort Project Alpha) : (Actual Effort Project Beta) = 4.58

This gives information that project alpha actually took about 4.58 times as many days to develop as the project beta. This information is useful for comparing with a similar ratio of POPs between the two projects.

Comparing the POP count for the two projects reveals project alpha is approximately four times as large, in terms of the POP count, compared to project beta, using the following percentage calculation:

$$\frac{POP\ \alpha}{POP\ \beta} = \frac{10478}{2566} \approx 4.08 \qquad (4)$$

Table VII shows the proposed refined POP Count values for the Projects Alpha and Beta.

TABLE VII. REFINED POP COUNT FOR PROJECTS

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

$$\frac{POP\ \alpha}{POP\ \beta} \text{Re}\ fined \approx 4.411 \qquad (5)$$

The result from equation (5) is more close to the Actual effort ratio (4.58) in comparison to the original POP Ratio from equation (4) hence further validate the proposed refinement in calculation of POP Count.

CONCLUSIONS

One of the factors that could affect the adoption of POP methods in practice is the lack of support tools to help estimators in their tasks. Another problem is the complicated formulation of POP Count. Here the simplified version of POP Count formula has been proposed. Thus POP Metrics calculations have been simplified. The refined POP Count values have been measured by an APA (Automated Predictive Object Point Analyzer) tool for various projects. The results were analyzed in terms of size and efforts. The suggested simplification in POP calculation showed more accurate results in terms of effort estimation and give more understandability of the system hence validate the proposed refined formula for POP metrics calculations. The automated tool after introduction of simplification in calculation makes it easy to use by practitioners.

Here an exhaustive range of projects have been analyzed. The projects covered during research presented more accurate results however the data to be studied may include additional projects in order to further ensure the validity for this refinement and hence accuracy of the measurement.

REFERENCES

[1] C. Ravindranath Pandian, (2005), "Software Metrics: A Guide to Planning, Analysis and application". Auerbach Publications A CRC Press Company Boca Raton London New York Washington, D.C.

[2] Arlene F. Minkiewicz, (1997), "Object- Oriented Metrics" Software Development. Wiley Computer Publishing, pp. 43-50. At: http://www.sdmagazine.com

[3] Haugh. M, E. W Olsen and Bergman. L. (2001), "Software Process Improvement: Metrics Measurement and Process Modelling", Vol 4, New York, Springer, pp. 159-170.

[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. 410-418

[5] CCCC Metric Tool by Tim Littlefair. http://www.fste.ac.cowan.edu.au/~tlittlef/

[6] 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.,

[7] Shyam R. Chidamber and Chris F. Kemerer, (1994)" A Metrics Suite for Object Oriented Design". IEEE transactions On Software Engineering, Vol. 20, No. 6, June 1994, pp. 476-493.

[8] B Boehm, C Abts, and S Chulani, (2000)."Software Development Cost Estimation Approaches - A Survey, Technical Report USC - CSE-2000-505", University of Southern California - Center for Software Engineering, USA, 2000.

[9] I. Sommerville, (2001)", Software Engineering", Sixth Edition. Addison- Wesley Publishers Limited, 2001.

[10] K.K Aggarwal and Yogesh Singh., (2006)", Software Engineering", Revised 2nd Edition, New Age International (P) Limited, Publishers, New Delhi.

[11] Alain Abran, (2010). "Software Metrics and Software Meterology", IEEE Computer Society, A John Wiley & Sons, INC., Publications.

[12] A. Abran and J. P. Jacquet, (1997). "From Software Metrics to Software Measurement Methods: A Process Model", 3rd Int. Standard Symposium and Forum on Software Engineering Standards (ISESS'97), Walnut Creek, USA.

[13] Goodman. P., (2004)."Software Metrics: Best Practices for Successful IT Management", Rothstein Associates Inc., Publisher At: www.rothstein.com

[14] Lorenz, M and Jeff Kidd., (1994). "Object- Oriented Software Metrics". A Practical Guide. New Jersey, Prentice Hall.

[15] Dr. Rakesh Kumar and Gurvinder Kaur (2011). "Article: Comparing Complexity in Accordance with Object Oriented Metrics". International Journal of Computer Applications, 15(8): February 2011, pp. 42–45.

[16] M. Xenos and D. Stavrinoudis and K. Zikouli and D. Christodoulakis.,(2000). "Object- oriented metrics – a survey", proceedings of the FESMA 2000, Federation of European Software Measurement Associations, Madrid, Spain.

[17] Caldiera, G. ; Price Waterhouse Coopers, Fairfax, VA, USA ; Antoniol, G. ; Fiutem, R. ; Lokan, C. , (1998)" Definitions and Experimental Evaluation of Function Points for Object Oriented Systems." Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International. 20-21 Nov 1998. Bethesda, MD., pp- 167 – 178.

[18] S.S Vicinanza, T. Mukhopadhyay., and M.J. Prietula, "Software-effort estimation: an exploratory study of expert performance", Information Systems Research, 2(4): 243 262, December 1991.