Automated Estimation of Predictive Object Points Metric Values for Object-Oriented code

Vijay Yadav, VibhashYadav, Raghuraj Singh

Abstract: ***If we are to improve the OO software quality we develop, we must measure our designs by well-defined standards. Possible problems in our system designs can be detected during the development process. If we are to estimate and manage our efforts, we must measure our progress effectively. An object-oriented coding scheme is, in some ways, incompatible with ancient metrics. To measure the size of software system, a variety of techniques for code development estimation exist like SLOC and Function Point, SLOC, as a metric has a number of drawbacks one is SLOC, is not consistent across languages, applications, or developing environments, however, it cannot applied on object-oriented code. The efforts required to develop the code is being calculated using the Predictive Object Point (POP) metric. This counting technique is supported by the Function Point Technique. This paper address the way of effort calculation for object oriented code using POP. To measure the POP accurately, an automated tool has been developed and designed. The results of the tools has also been discussed in the paper.***

Index Terms**:** ***Code Estimation Technique, Software Tools, System Metric Tools, Object-Oriented Metrics, Predictive Object Point Metric, Automation, and improvement.***

I. INTRODUCTION TO SOFTWARE METRIC AND MEASUREMENT

Metrics thought to be used for both i.e. to search out anomalies and to measure the progress of the software development, the metrics can be considered as tips and not as the rules.

The metrics will facilitate to support the required motivation in software system development method [11]. The Metric is claimed to be as a standard of the estimation process. Used to judge the features of the measured software system, like development size and effort, complexity, quality in an efficient manner. Measuring the incorrect factor serves no purpose.

Most of the people would agree that source Lines of code (SLOC) have very little or no meaning, because of variation in types of language, code complexity, and reuse. We have a tendency to should concentrate on what we are trying to attain, we want as little code as necessary to fulfill our functional necessities. This could lead to low price and development time for that function [10].

Estimation is thought of as the mapping from the experimental world to the recognized, relational world. It's the quantity or symbol distributed to an element by this mapping strategy to describe a trait [20].

Measurements are identified with upgrades. The worldview ended up being "in the event that you can't quantify, you can't improve" [6]. Inside the logical fields, together with building, estimation is viewed as one of the quantities of diagnostic apparatus and depends on a huge collection of information planned up over hundreds of years. [1], [2].

The prologue to programming framework measurements accompanied work move from a conventional to production service programming framework designing environment [12]. Moving from estimations to measurements resembles moving from perception to comprehension [8]. They offer an indication of the advancement that a task has made and furthermore the nature of the plan. Measurements are markers and encourage in making information driven decisions in time [11].

By the execution of OO paradigm, the analysts changed and approved the conventional measurements in principle or experimentally. Measuring and unpredictability measurements were the most amazing commitments for size and effort estimation in OO setting [7].

Most methodologies for assessing effort need an estimation of the size of the software system. Once a size estimate is accessible, models that depicts size to the effort will be utilized [14].

II. DESCRIPTION OF POP METRIC

Mickiewicz proposed POP metric in 1998. Price Systems [3] has built up the predictive Object point (POP) metric for anticipating the effort requiredfor building up an object-oriented programming framework. POPs are implied as an improvement over FPs, that was initially intended to be utilized among procedural frameworks, by illustration on understood measurements identified with an object-oriented framework [9]. It had been structured explicitly for an Object-oriented programming framework and result from the estimation of the object-oriented properties of the framework. It satisfied most of the components of OO ideas and depended on the calculation procedure of function point (FP) procedure [3].

A. What metrics include in POP code Sizing Metric?

With the use of the Object-Oriented Paradigm, the researchers changed and valid the standard metrics on paper or through empirical observation [3]. The following metrics are involved in object-oriented framework in POP Count: Number of top-level classes (TLC), Average number of weighted methods per class (AWMC), Average depth of inheritance tree (ADIT), and Average number of children per base class (ANOC) [16]. AWMC, ADIT, and ANOC are assembled from the MOOSE metrics suite [15].

Revised Manuscript Received on July 22, 2019.

Vijay Yadav, C.S.E, A.K.T.U, Lucknow, India.
VibhashYadav, I.T, R.E.C, Banda, India.
Raghuraj Singh, C.S.E , H.B.T.U, Kanpur, India.

B. POP Count Formula

The preceding metrics are then collected as planned by Minkiewicz (1997)Refer to "(1),", giving the number of POPs required to predict the software size.

$$f1(a,b,c) = a * (1 + ((1+b) * c)^{1.01} + (|b-c|)^{.01}) \quad (1)$$
$$f2(b,c) = 1.0$$
$$POPs(a,b,c,d) = \frac{d * f1(a,b,c)}{7.8} * f2(b,c)$$

Here the parameters for the POP formula can be taken a= TLC(Top Level Class), b=NOC(Number of children), c=DIT(Depth of Inheritance Tree, d=WMC(Weighted Method per Class) where f1 predicts the size the full OO software code, and f2 gives the effects of reuse through inheritance [4].

III. POP COUNT IMPROVEMENT SUGGESTED

The utilization of POP technique couldn't increase enough acknowledgement from specialists to be utilized on a usual owing to the shortage of an easy to use support tool and too several tough formulations. POPs use truly, was unfeasible because of it required completely an unnecessary amount of detail for top-down estimation [14].
Predictive Object Points are created inside the setting of organizations(for example Price Systems). In any case, no insights about their genuine use in these organizations are in open available [13]. This investigation was led about twenty years past and unfortunately has been no longer in real use [3], [14].
The tool Automated POPAnalyzer (APA)is utilized for evaluating POPs and changes are applied in APA tool for approving the results[14].

Jain et al., (2014) has proposed the refined formula for POP Metric calculation, Refer to "(2),".

$$f1(a,b,c) = a * (1 + ((1+b) * c)^{1.01}) \quad (2)$$
$$f2(b,c) = 1.0$$
$$POPs(a,b,c,d) = \frac{d * f1(a,b,c)}{7.8} * f2(b,c)$$

Here the parameters for the different OO metrics are a=TLC, b=NOC, c=DIT, d=WMC respectively. Simplified POP count formula has already been suggested and validated [14].

IV. ESTIMATION OF OO PROJECTS THROUGH POP

The POP metric estimation especially relies upon the object-oriented structure design. Fuse of ongoing classes that don't have any parent among the framework may result in the change in design plan. POP count value depends on the TLC if such top-level classes are less in range, the WMC (Weighted Method Count) worth will also decrease and so reducing the overall POP count value [17]. The WMC Metric Value which is hard to compute is already simplified by Yadav et al (2016)a by presenting another OO Metric AWC (Average Weighted Method Count). The methods were divided into various types according to dimensions suggested by Minkiewicz (1997) over the manual examination of code. In any case, Judge and Williams, (2001) have portrayed in his paper that any business would be profited by utilizing a part dependent on its own past data. Inside a similar methodology, the strategy types are regularly characterized into complexity types. The outcome will be better when a high number of pure abstract classes are connected that is valid for JAVA code . The interfaces can be assumed of as pure abstract classes.

Explanation behind the count improvement is extensively to the number of classes which are high-level. With an ascent in TLC, a POP no. will increment. This is frequently not valid for ventures dependent on C++ that will in general utilize pure abstract classes less broadly.

In evident OO setting, a framework ought to be partitioned into numerous subsystems and each subsystem may be separated into numerous phases as indicated by time span. The refinement may be considered for the use of POP [16].

In this manner frameworks must be part into components and additional partitioned into sub-modules.

So the POP no. of every C++ file will be precisely determined based on its individual C++ document which gives better outcomes to the general estimation of POP.

The accompanying procedure was proposed by Judge and Williams (2001) pursued for the estimation count of POP:

A. Step 1

The initial step was to find out the SLOC metric for OO projects through APA tool dependent on CCCC, an OO metric finding tool. [3].

B. Step 2

The average DIT(ADIT) metric can be calculated by the DIT metric got from CCCC tool for every class file,Refer to "(3),".

In the same way the average NOC(ANOC) can be find out from the NOC metric calculated for every class file from CCCC tool, Refer to "(4)," [13],14],[18].

$$\text{ADIT} = \frac{\text{Sum (DIT Classes)}}{\text{Row Sum (NOC +Positive DIT Count)}} \quad (3)$$

$$ANOC = \frac{\text{Sum (Base Class NOCs)}}{\text{Positive NOC Count Classes}} \quad (4)$$

C. Step 3

The method value is divided by the class value to get the Average Method count (AMC) [16]. what's more, this ought to be under 20. High value represent the higher responsibility in less number of classes [10].

D. Step 4

And then the TLC metric for each C++ file is then determined. This having the classes without any guardians or parents and are at top of all other classes.

TLC metric are at the level 0 (root) in the class diagram and every single different classes are gotten from it [14].

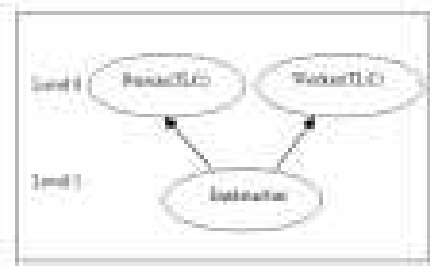

Figure 1: Flow Diagram of Person.cpp.

Average DIT= ((2*0) + (1*1))/3=0.33
Average NOC= ((1*1) + (1*1)/2=1
TLC=2.

On running of the project Person.cpp shown in "Fig." 1, on our tool gives similar results as shown in "Fig." 2. The outcomes are shown as the enclosed value in the print.

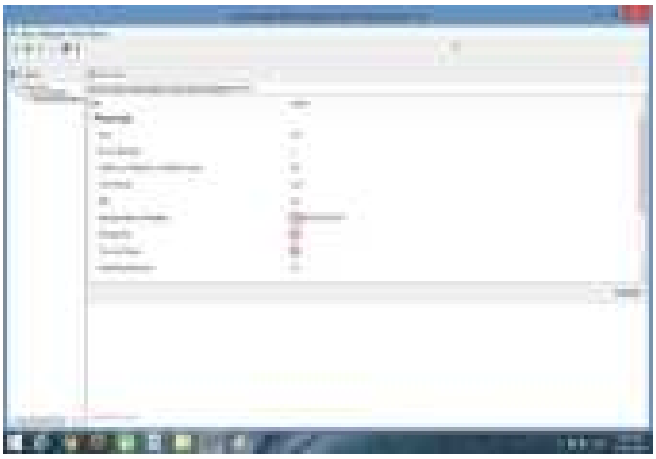

Figure 2: TLC Values, Metric Avg DIT, Avg NOC

E. Step 5

After gathering all the required metrics, WMC is calculated. We will apply the weight age to calculate the count of average methods. The calculation for average count are as follows [3]:

AC/DMC = 20% of AMC
ASMC = 30% AMC
AMMC = 45% AMC
AIMC = 5% AMC

Where AMC is Average Method per Class, AC/DMC is Average Constructor or Destructor Method Count, ASMC is Average Selector Method Count, AMMC is Average Modifier Method Count, AIMC is Average Iterator method Count.

There are three complexities count of methods using weighting method:

LCMC = 22% AMC
ACMC = 45% of AMC
HCMC = 33% of AMC

Where LCMC is Low Complexity Method Count, ACMC is Average Complexity Method Count, HCMC is High Complexity Method Count and AMC is Average Method Count.

A total of twelve calculations were performed for each C++ file and WMC given using their sum [16]. For WMC calculation for the overall project the same method can be utilized.

V. DESCRIPTION OF EMPIRICAL STUDY

3 C++ projects together with a pair of projects from the analysis work of Judge and Williams (2001) for validating the POP metric are thought of for this study.

Table I: C++ projects to find metrics in POP estimation

Project No.	Project Name	SLOC	TLC
1.	Percentage.cpp	87	2
2.	Person.cpp	72	2
3.	Rectangle.cpp	46	2

(see Table II) shows the metrics like NOM, classes, AMC, WMC.

Table II: Projects Values of Metrics.

Project No.	Methods	Classes Which have method count	AMC	WMC
1.	8	5	1.6	16.7648
2.	10	4	2.5	26.195
3.	7	4	1.75	18.3365

Execution of the project Rectangle.cpp shown in (see Table I) gives the same result as shown in "Fig." 3, on the APA tool. The metric values are shown as the circled value in the figure.

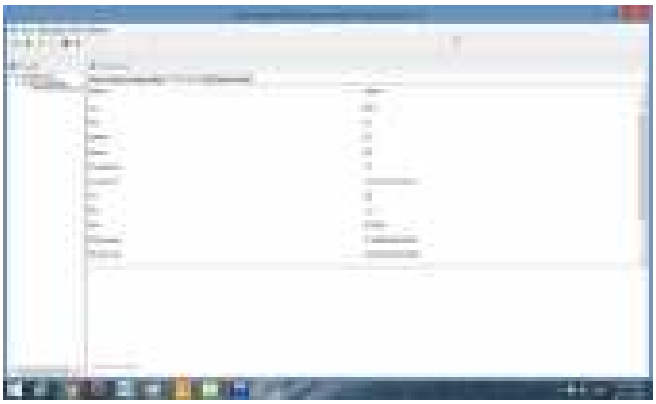

Figure 3: Sample SLOC, Methods, Classes, TLC and WMC Values through APA Tool.

The effort for correlation with Refined POP and Original POP has been determined over the COCOMO model that was proposed by Boehm et al. in 1981, (2000).

It's normally referenced as COCOMO 81 model for the development recognition, this model expect that the software size will be determined in thousands of delivered source instructions (KLOC) and so uses a non-linear equation to predict the effort for the projects [5].

The formula is given by Refer to "(5),".

$$\text{Effort} = a*(\text{KiloLOC})^{b} \text{ Person Month} \quad (5)$$

The constraints a and b are the individuals who hold the intrigue. The idea is to plot them dependent on the qualities of the venture .

These attributes are then contrasted with chronicled data to give the right value for these constraints[19].

This is completely simple model. Boehm's et al., (2000) thought of 3 modes of software system development in this model: organic, semi-detached and embedded [5].

Table III: Comparison of Effort values with Original POP and refined POP

Proj. No.	POP_{Ori}	POP_{Ref}	Effort Values
1	15.0123	10.7729	0.1848
2	17.866	11.18	0.1515
3	12.51	7.8234	0.095

On examination of two projects, the POP count says P1/P2 reveals how much larger the project P1 is of project P2 in terms of size of code like POP [16].

For various combinations of projects, this POP count ratio has been calculated using both the POP_{Ori} Count value and POP_{Ref} Count Values (see Table III).

Each ratio is compared with the effort ratios for similar projects.

The refined and Original POP values are reflected in "Fig." 4.

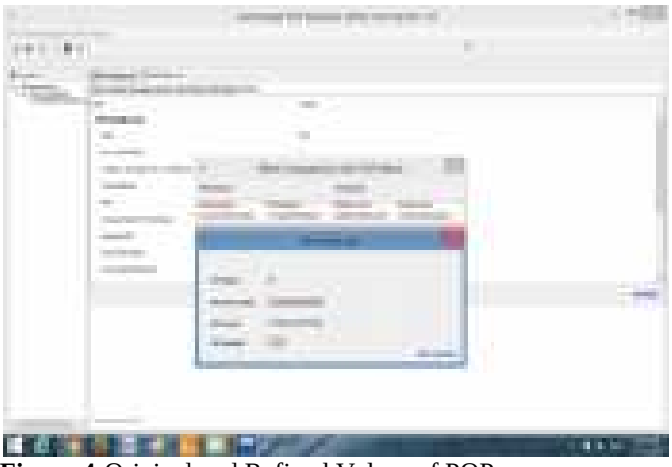

Figure 4: Original and Refined Values of POP

The Judge and Williams (2001) in his paper proposed that closer this ratio to that of the ratio of the effort, the more correct the POP technique is.

This is as a result of Effort and POP count are proportional, wherever the proportionality constant is that the POP productivity rate. The results are given in (see Table IV).

Table IV: Ratio of SLOC, Original POP, Refined POP & Effort.

S.No	Project Compared	SLOC Ratio	POP_{Ori} Ratio	POP_{Ref} Ratio	Effort Ratio
1.	P1| P3	1.8913	1.2000	1.3770	1.9453
2.	P2| P3	1.5652	1.4281	1.4290	1.5947

The refinement proposed in the POP formula can be checked in reference to the projects taken by Judge and Williams (2001).

In their analysis work using project Alpha and Beta, they established POP metric as an improved indicator of software size compared to Function Point metricas shown in (see Table V) [16].

Table V: Project Metrics for Alpha and Beta [16].

Project Attributes	Project Alpha	Project Beta
SLOC	38854	20570
Total Number of Methods	2412	833
Total Number of Classes	404	147
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
TLC(Top Level Class)	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

The effort metrics has been developed along with the developers effort to find the actual time or effort spent on the code development.

The project effort statistics are presented as follows [16]:

($Effort_{Actual}$ of α Project) : ($Effort_{Actual}$of β Project) = 4.58

From the above ratio, we can conclude that project α takes about 4.58 times more effort than project β [14].

Distinguishing between the POP value of the two projects reveals project α is just about four times as big, in terms of the POP value, in comparison to project β, using the subsequent percentage calculation as shown in Refer to "(6)," [16] :

$$\frac{POP\alpha}{POP\beta} = \frac{10478}{2566} \approx 4.08 \quad (6)$$

(see Table VI) depicts the $POP_{Refined}$ values proposed for the α & β Projects [14].

Table VI: Project Alpha & Beta Refined POP count Values[14]

POP Refined Value for Projects	α	β
No. of $POP_{Refined}$	8849.422	2006.1515

$$\frac{POP\alpha}{POP\beta} \text{Re } fined \approx 4.411 \quad (7)$$

This can be seen that the outcome from Equation (7) is nearer to the particular $Effort_{Actual}$ ratio of α & β Projectsas seen from the original POP ratio from Equation (6) thus shows the validation of the estimation ofPOP Count in C++ projects.

VI. CONCLUSION

Taking everything into account, the POP metric when connected to past information on two of Parallax's deployed projects gave an improved sign of their size than did the past POP count values. POP is a metric, that contemplates some well-characterized metrics inside an object-oriented system. However, it's difficult to develop the OO design for the complete system in an industrial setting. Cautious plan and examination are required for advanced subsystems. This implies that the POP metric may well be best applied to the estimation of the C++ projects which may be additionally classified into essential subsystems.

This study therefore not solely helped to induce some understanding of the systems however additionally proven that POP an improved effort predictor. However, still more number of projects under various categories will be taken for analysis in order to ensure validity. Another future study prospect would be to benefit the analysis of historical code by incorporating the McCabe's Cyclomatic Complexity Metric (MVC) with the POP Metric. Complexity analysis may be done with some appropriate analysis techniques. Risk analysis of the various projects is additionally taken into consideration for another future study prospect.

ACKNOWLEDGMENT

I am feeling a great sense of self-satisfaction and good experience having accomplished my Ph.D. research paper entitled “Automated Estimation of Predictive Object Points Metric Values for Object-Oriented code”. I would like to thanks Dr. VibhashYadav (Associate Professor and Head), Department of Information Technology, Rajkiya Engineering College, Banda, my Supervisor for his guidance, support, motivation ,and encouragement throughout the period this work was carried out. His readiness for consultation at all times, his educative comments, his concern , and assistance even with practical things have been invaluable.

I would like to express special thanks to Prof. Raghuraj Singh (Prof. H.B.T.U Kanpur) my research guide who have contributed their precious time and effort to help me. Without whom it would not have been possible for me to understand and complete my paper.”.

REFERENCES

1. Alain Abran, *Software Metrics and Software Meterology*, IEEE Computer Society: A John Wiley & Sons, INC., Publications, 2010.
2. A. Abran and J. P. Jacquet, "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, 1997.
3. Arlene F. Minkiewicz.(1997), "*Object-Oriented Metrics" Software Development.* Wiley Computer Publishing, pp. 43-50. Available: http://www.sdmagazine.com.
4. Arelen F. Minkiewicz, "Measuring Object-Oriented Software with Predictive Object Points", Project Control for Software Quality, A. C. Rob Kusters, Fred Heemstra and Erik van Veenendaal, Ed.: Shaker Publishing, 1999.
5. B Boehm, C Abts, and S Chulani, ”Software Development Cost Estimation Approaches - A Survey, Technical Report USC -CSE-2000-505”, University of Southern California - Center for Software Engineering, USA., 2000.
6. C. Ravindranath Pandian, "Software Metrics: A Guide to Planning, Analysis, and application". *Auerbach Publications A CRC Press Company*, Boca Raton London New York Washington, D.C, 2005.
7. Dr. Rakesh Kumar and Gurvinder Kaur, "Article: Comparing Complexity in Accordance with Object-Oriented Metrics". *International Journal of Computer Applications*, vol.15(8), pp. 42–45, Feb. 2011.
8. Goodman. P., (2004).*Software Metrics: Best Practices for Successful IT Management*, Rothstein Associates Inc., Publisher, Available: www.rothstein.com.
9. Haugh. M, E. W Olsen, and Bergman. L,“Software Process Improvement: Metrics Measurement and Process Modelling”,*Springer*, vol. 4, New York, pp. 159-170, 2001.
10. Mark Lorenz.,"*Object-Oriented Software Development: A Practical Guide*" ., ISBN 0-13-1726928-5.,Prentice Hall Ptr Upper Saddle River, New Jersey 07458, 1993, 227 p.
11. Mark Lorenz and Jeff Kidd.,"*Object-Oriented Software Metrics: A Practical Guide*" ., ISBN 0-13-179292-X., Prentice Hall Englewood Cliffs, New Jersey 07632.,1994, 146 p.
12. M. Xenos and D. Stavrinoudis and K. Zikouli and D. Christodoulakis.,"Object-oriented metrics – a survey",*Proceedings of the FESMA 2000, Federation of European Software Measurement Associations*, Madrid, Spain, 2000.
13. Shubha Jain, Vijay Yadav, and Prof. Raghuraj Singh, "OO Estimation Through Automation of Predictive Objective Points Sizing Metric"., *International Journal Of Computer Engineering and Technology (IJCET)*, vol. 4, (Issue 3), pp. 410-418, May- June. 2013.
14. Shubha Jain, Vijay Yadav, Prof. 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 I*SBN 978-1-4799-2571-1, IEEE Computer Society, Gurgaon, India,Feb 21st-22nd, 2014.
15. Shyam 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.
16. T. R Judge, A. Williams, “OO Estimation – an Investigation of the Predictive Object Points (POP) Sizing Metric in an Industrial Setting”. Parallax Solutions Ltd, Coventry, UK.,2001.
17. Vijay Yadav, Dr. Vibhash Yadav, Prof. Raghuraj Singh., “Introducing New OOMetric For Simplification In Predictive Object Points (POP) Estimation Process In OO Environment”., *International Journal Of Engineering Sciences & Research Technology (IJESRT)*, ISSN: 2277-9655 vol. 5 (Issue 1), , pp: (716-723),Jan. 2016 Impact Factor: 3.785,

18. Vijay Yadav, Dr. Vibhash Yadav, Prof. Raghuraj Singh.,(2016). "Identifying The Relationship Between Original And Refined POP Metric For OO Software*"., International Journal Of Research Fellow For Engineering (IJRFE)*, ISSN: 2320-7396 vol. 4 (Issue 11), pp. 39-46,Nov-2016 Available:www.ijrfe.com.

19. Vijay Yadav, Dr. Vibhash Yadav, Prof. Raghuraj Singh.,"An Automated Software Cost and Schedule Measurement Tool for OO System using Predictive Object Point Sizing Metrics"., *International Journal Of Emerging Technologies & Innovative Research (IJETIR)*,ISSN: 2349-5162,UGC Approved, Impact Factor:5.87, Published in vol. 5 (Issue 2) , February-2018, pp. 1106-1111, Nov. 2016,Available: www.jetir.org. .

20. Vu Nguyen," Improved Size and Effort Estimation Models For Software Maintenance", Ph.D. Dissertation, University Of Southern California, 2010.

AUTHORS PROFILE

Vijay Yadav He is B.TechHons in (I.T) from C.S.J.M University Kanpur (U.I.E.T) and M.TechHons in (C.S.E) from U.P Technical University. Currently, he is doing a Ph.D. in (C.S.E) from Dr. A.P.J. Abdul Kalam Technical University, Uttar Pradesh. He has undergone projects like Online Entertainment world, Enterprise Resource Planning System during his B.Tech (I.T) curriculum. He has done his M.Tech (C.S.E) thesis in Object-Oriented Software Metrics (POP Automation). He is a meritorious student of B.Tech (I.T) and M.Tech (C.S.E) and has got merit excellence award. He has 8 papers in International Conferences/Journals and two IEEE conference attended for paper presentation.

VibhashYadavHe is a B.Tech. (CSE) from C.S.J.M University, Kanpur, M.Techin (C.S.E) from MaharshiDayanand University, Rohtak and Ph.D. in Computer Science and Engineering from Uttrakhand Technical University, Dehradun. He has about 10 years of experience in teaching. Currently, he is Associate Professor and Head in I.T Dept of Rajkiya Engineering College, Banda. He has guided 4 M.Techs and several B.E./B.Tech projects. He is the Member, Kanpur Chapter, CSI. He has more than 15 papers in National / International Conferences and Journals to his credit. Currently, 4 students are working for Ph.D. and 4 are pursuing M.Tech. under his guidance.

Raghuraj Singh He is a B.Tech. (CSE), M.S. (Software Systems) and Ph.D. in Computer Science and Engineering from U.P. Technical University. He has about 23 years of experience in teaching. Currently, he is HOD C.S.E in HBTU Kanpur. He has guided 7 PhDs and 17 M.Techs and several B.E./B.Tech projects. He is the Chairman, Kanpur Chapter, CSI, Life Member of ISTE, Member of the Institution of Engineers (India), Fellow Member of IETE, Professional member of ACM and Senior Member of International Association of IACSIT. He has more than 80 papers in National / International Conferences and Journals to his credit. Currently, 4 students are working for Ph.D. and 4 are pursuing M.Tech. under his guidance.