ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

INTRODUCING NEW OO METRIC FOR SIMPLIFICATION IN PREDICTIVE OBJECT POINTS (POP) ESTIMATION PROCESS IN OO ENVIRONMENT

Er. Vijay Yadav*, Dr. Vibhash Yadav, Prof. Raghuraj Singh
*Department of Information Technology, Government Girls Polytechnic Charkhari, Mahoba(India)
Department of Computer Science & Engineering, Krisha Institute of Technology, Kanpur (India)
Department of Computer Science & Engineering, Kamla Nehru Institute of Technology, Sultanpur (India)

ABSTRACT
Many effort estimation techniques exist for sizing software systems but none is directly used to measure object-oriented software. Most of the researchers have worked for size and effort evaluation but still the problem has not been fully resolved. Many of the existing estimation 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 lack 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, we have developed simple formulations for POP calculation. The POP count formula suggested by PRICE system for estimating the effort has been simplified by introduction of new OO metric named AWC(Average Weighted Method Count). AWC metric 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.

KEYWORDS: Object Orientation, Software size Measurement, Software Metrics, Predictive Object Point, Automation, Average Weighted Method Count.

INTRODUCTION TO MEASUREMENT
With the quick growth in software industries, corporate developers faced an interesting variance between two emerging trends: Object Oriented development and metrics.
They found that object oriented technology is, in many ways, inconsistent with traditional metrics. Good measurement program is the choice of good metrics. 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 [5]. Traditional design techniques separate data and procedures while object-oriented designs combine them.
It is important to measure the amount of raw functionality the software delivers, but it is equally important to include information about communication between objects and reuse through inheritance in the 'size' as well [1]. Measurements are associated with improvements. The paradigm turned out to be "if you can't measure, you can't improve" [2]. In the scientific fields, including engineering, measurement is considered to be as one of the number of analytical tools and is based on a large body of knowledge built up over centuries [6] [7]. Most methods for estimating effort require an estimation of the size of the software. Once a size estimate is available, models that relate size to effort can be used [4].
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 [3]. Metrics are 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 [10] [11].

OVERVIEW OF 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. 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. POPs are intended as an improvement over FPs by drawing on well-known metrics associated with an object oriented system [13]. POPs are suitable metrics for estimating the size and subsequently the effort required for development of object oriented software [15], based on the behaviors that each class is delivering along with top level inputs describing the structure of a system [9].

What Contribute to POP Software Sizing Metric?

By the implementation of Object Oriented Paradigm the researchers modified and validated the conventional metrics theoretically or empirically [11]. The following metrics measure 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 [12][14].

How to Calculate POP Count?

The above mentioned metrics are then gathered to form the equation (1), giving the number of POPs for a system [8].

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

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

SUGGESTED SIMPLIFICATION IN POP COUNT CALCULATION

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 [9]. 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 $|NOC\text{-}DIT|^{.01}$ 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).

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

In this paper 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 which involves very rigorous method of calculation. This can be used for java projects. However this may not be true for other environments.

POP COUNT CALCULATION PROCESS

The following process was followed for calculation of POP Count:

Step 1

The first step was to obtain the Source Lines of Code (SLOC) metric for projects through APA tool [8] based on CCCC, an object oriented metric gathering tool [9].

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).

Step 3

Average Method count (AMC) is calculated by dividing the method count by the class count [12].

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].

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 [2]:

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.

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.

DESCRIPTION OF EMPIRICAL STUDY

The proposed simplification in WMC calculation for POP count formula for Java Projects can be validated ,under this study, 24 projects including 2 projects from research work of T. R Judge and A. Williams [16] as shown in Table 1 have been considered.

Table 1. Projects analyzed to study simplification in POP Calculation

Project No.	Project Name	No. of Java Files	SLOC	TLC
1.	Face_Detection_Syst	3	600	2
2.	JaimLib_Ver_0.4	45	1505	44
3.	JaimLib_Ver_0.5	45	1539	44
4.	Mobile_Pay_Service	32	2887	47
5.	Online_Address_Book	12	614	12
6.	PhysicsMata_ver_0.3	8	239	7
7.	PhysicsMata_ver_0.3.1	8	237	7
8.	PhysicsMata_ver_0.5.0	11	398	9
9.	PhysicsMata_ver_0.5.1	12	416	9
10.	PhysicsMata_ver_0.5.2	34	1168	29
11.	PhysicsMata_ver_0..6.0	5	224	6
12.	PhysicsMata_ver_0..6.1	5	227	6
13.	PhysicsMata_ver_0.8.0	12	227	7
14.	PhysicsMata_ver_0.8.1	10	230	7
15.	PhysicsMata_ver_1.2.0	19	418	15
16.	PhysicsMata_ver_1.2.1	19	419	15
17.	JavaGeom_ver_0.3.0	53	2687	57
18.	JavaGeom_ver_ 0.3.2	55	3179	60
19.	JavaGeom_ver_0.5.0	67	2888	80
20.	JavaGeom_ver_ 0.5.2	78	3311	93
21.	JavaGeom_ver_0.5.1	72	2868	80

22.	JavaGeom_ver_ 0.6.0	74	3190	91
23.	JavaGeom_ver_0.8.0	104	3869	134
24.	Lwjgl_0.92	266	18262	96

Table 2 shows the number of methods, classes, AMC, WMC needed for POP Count calculations for all chosen projects.

Table 2. Metric values for chosen projects

Project No.	**Methods**	**Classes with+ve method count**	**AMC**	**WMC**	**AWC= WMC/AMC**
1.*	30	3	10.0	104.78	10.478
2.*	207	45	4.6	48.199	10.478
3.	210	45	4.667	48.897	10.477
4.*	152	32	4.75	49.771	10.478
5.	40	12	3.33	34.9267	10.488
6.*	20	4	5.0	52.39	10.478
7.*	20	4	5.0	52.39	10.478
8.*	56	10	5.6	58.677	10.478
9.*	59	11	5.3636	56.20	10.478
10.	144	27	5.33	55.883	10.484
11.*	36	4	9	94.302	10.478
12.*	37	4	9.25	96.9215	10.478
13.*	42	7	6.0	62.868	10.478
14.*	33	6	5.5	57.629	10.478
15.*	33	15	2.2	23.052	10.478
16.*	33	15	2.2	23.052	10.478
17.	495	50	9.9	103.711	10.475
18.	531	52	10.21	107.039	10.483
19.	533	58	9.1897	95.9564	10.441
20.	616	62	9.9355	103.7494	10.442
21.*	530	56	9.4364	98.874	10.478
22.	592	62	9.5484	99.7128	10.443
23.	654	76	8.6053	90.8585	10.558
24.	2418	276	8.9542	93.8424	10.480

The Project marked with * in corresponding S. No. give similar value of AWC 10.478.

Execution of the project PhysicsMata_ver_0.5.1 shown in Fig.1.1 on APA(Automated POP analyzer) tool, this tool is developed and automated by us for analyzing the POP metric, The results are highlighted as under the rectangled value in the snapshot. The value of the AMC for this project is found to be 5.3636 and the vaue of WMC is 56.20 as shown in Fig 1.1.

On analyzing the above projects, it was found that the value of the AWC metric most of the times comes out to be nearly 10.478 for all projects. Thus the value of AWC metric may be considered as 10.5 and the value of the WMC can be calculated as below equation:

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

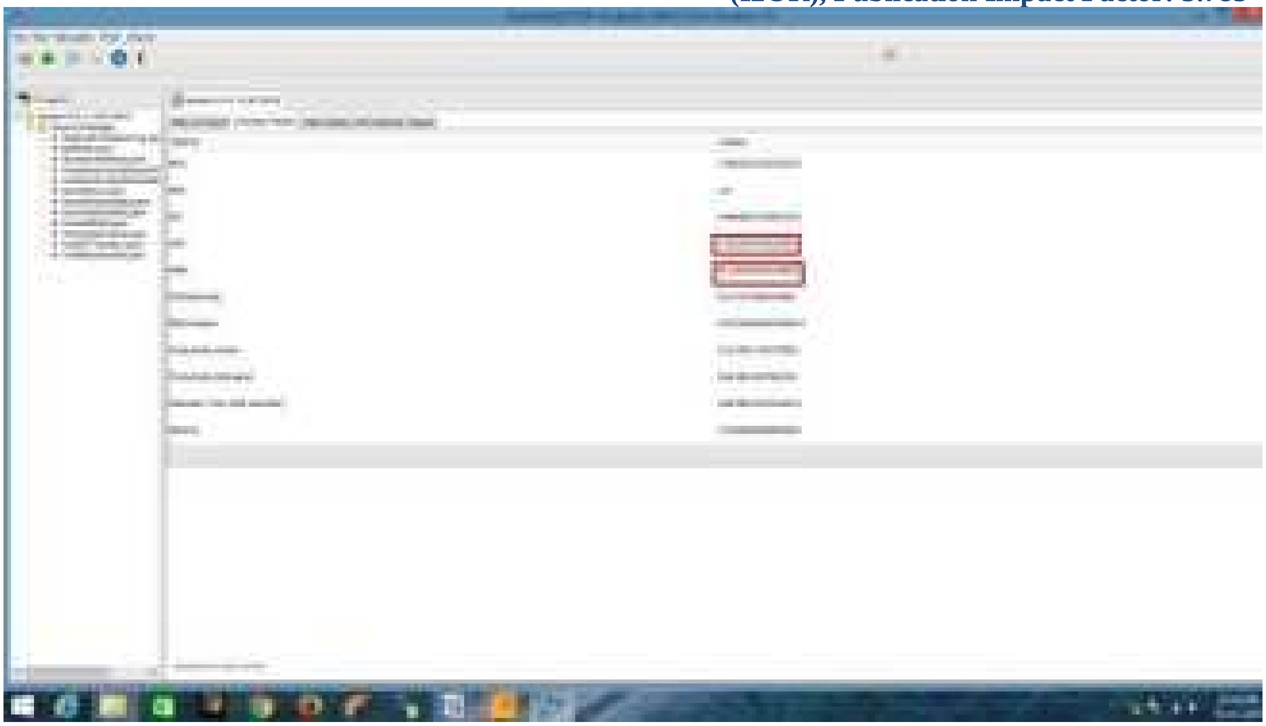


Fig 1.1: Sample AMC,WMC Values through APA Tool

And now on substituting the equation (3) to equation (2) we get the improved and simplified formula for refined POP calculation , as we know that from Step 5 for calculation of the WMC metric it is very tedious task to calculate it by various method complexities. Now the Simplified POP count formula is reduced to equation (4). Here for the calculation of WMC metric we just have to mutiply the value 10.478 with the AMC value.

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

$$POPs(WMC, NOC, DIT, TLC) = \frac{AMC * 10.478 * f1(TLC, NOC, DIT)}{7.8} \quad (4)$$

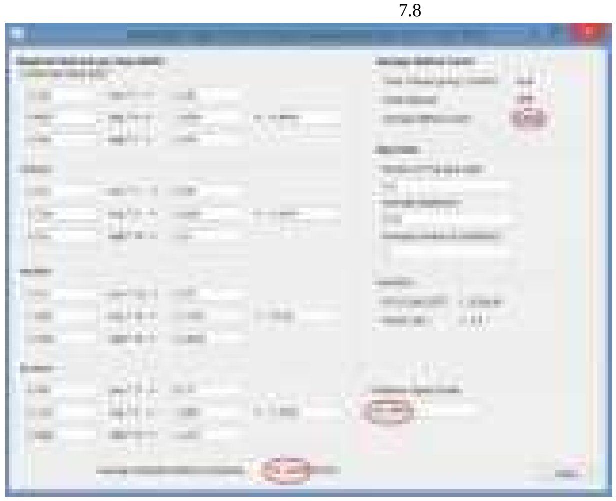


Fig 1.2: Sample AMC,WMC and Refined POP values for overall Project through APA Tool

Here from the Fig 1.2 it is clearly seen that the process of calculation of the WMC is very tedious, and the value of AMC, WMC and Refined POP for overall project PhysicsMata_ver_0.5.1 is also shown as circled in Fig 1.2.

RESULTS AND DISCUSSION

The proposed simplification in POP formula can further be checked in reference to the projects taken by T. R Judge and A. Williams [16]. 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 3.

Table 3. Summary of Project metrics [16]

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

Table 4 shows the value of the metric AWC which is calculated by dividing WMC metric with AMC metric for the Projects Alpha and Beta.

Table 4. AWC metric value for projects

Project Attributes	Project Alpha	Project Beta
AWC=WMC/AMC	10.478	10.478

The result from Table 4 gives the same value of the AWC metric (10.478) as we obtained in the Table 2 for the projects we investigated, hence this further validates the proposed simplification in calculation of the POP count by replacing WMC metric with the equation (3).

CONCLUSION

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 simplified version of POP count formula has been proposed through which POP metrics calculations have been simplified by replacing WMC metric which involves very tedious method of calculation. The projects taken for empirical study from research work of T. R Judge and A. Williams [16], presented same results as we proposed, however the data to be studied may include additional java projects. This will further ensure the validity for this simplification for Java Projects and hence accuracy of the measurement.

ACKNOWLEDGEMENTS

I am feeling great sense of self-satisfaction and good experience having accomplished my Ph.D research paper entitled “Introducing New OO Metric for Simplification in Predictive Object Points (POP) Estimation Process in OO Environment”. I would like to thanks Dr. Vibhash Yadav (Associate Professor and Head), Department of Computer Science & Engineering, Krisha Institute of Technology, Kanpur, 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.

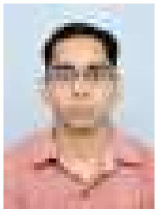
I would like to express special thanks to Prof.Raghuraj Singh (Director, KNIT, Sultanpur) my research guide who have contributed their precious time and effort to help me. Without whom it would not have been possible for me to

understand and complete my paper.".

REFERENCES

[1] A.F. Minkiewicz, "Measuring object-oriented software with predictive object points', Proc. ASM'97-Applications in Software Measurement, Atlanta.
[2] 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.
[3] Lorenz, M and Jeff Kidd., "Object- Oriented Software Metrics". A Practical Guide. New Jersey, Prentice Hall, 1994.
[4] Caldiera, G. ; Price Waterhouse Coopers, Fairfax, VA, USA ; Antoniol, G. ; Fiutem, R. ; Lokan, C. ,"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, 1998.
[5] Lorenz Jeff Kidd," Object Oriented Software Metrics : A Practical Guide, Prentice Hall,
[6] Alain Abran,"Software Metrics and Software Meterology", IEEE Computer Society, A John Wiley & Sons, INC., Publications, 2010.
[7] 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.
[8] Arlene F. Minkiewicz, "Object-Oriented Metrics" Software Development. Wiley Computer Publishing, pp. 43-50,1997 at: http://www.sdmagazine.com
[9] 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).
[10] Dr. Rakesh Kumar and Gurvinder Kaur, "Article: Comparing Complexity in Accordance with Object Oriented Metrics". International Journal of Computer Applications, 15(8): February 2011, pp. 42–45.
[11] 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.
[12] 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).
[13] 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.
[14] 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.
[15] Shubha Jain, Vijay Yadav, Raghuraj Singh., " An Approach for OO Software Size Estimation using Predictive Object Point Metrics", INDIACom-2014; 8th INDIACom-2014; 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 in IEEE Xplorer).
[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.

AUTHOR BIBLIOGRAPHY



Er. Vijay Yadav

He is B.Tech Hons in (I.T) from C.S.J.M University Kanpur (U.I.E.T) and M.Tech Hons in (C.S.E) from U.P Technical University. Currently he is working as Lecturer I.T in Government Girls Polytechnic Charkhari (Mahoba). Technical Education Department Government.U.P. Currently he is doing 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 meritorious student of B.Tech (I.T) and M.Tech (C.S.E) and have got merit excellence award. He has 5 papers in InternationalConferences/Journals and two IEEE conference attended for paper presentation.

Dr. Vibhash Yadav

He is B.Tech. (CSE) from C.S.J.M University,Kanpur, M.Tech in (C.S.E) from Maharshi Dayanand 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 in Department of Computer Science & Engg. at KIOT, Kanpur. 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 PhD and 4 are pursuing M.Tech. under his guidance.

Prof. Raghuraj Singh

He is 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 Director. at KNIT, Sultanpur. 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 PhD and 4 are pursuing M.Tech. under his guidance.