ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Survey on Defect Prediction and its Effect on Software Quality

Prabujeet Kaur, Dharmendra Lal Gupta

M.Tech Student, Dept of Computer Science, Kamla Nehru Institute of Technology, Sultanpur, UP, India

Associate Professor, Dept of Computer Science, Kamla Nehru Institute of Technology, Sultanpur, UP, India

ABSTRACT: Many of the researchers in their studies have given their unique ways to predict the model behavior for a specific dataset in order to identify the defective modules and faulty modules in a class and to improve the quality of the software. The main aim of this paper is to give an overview of some of the papers and summarize their work so that the researchers may find it easy for them to identify and evaluate the current work that has been in trend today. The study includes the survey of year 2015 and 2016. This will help the researchers to channelize their work in a specific direction.

KEYWORDS – defect prediction; object oriented metrics; performance measures; classifier or learner or machine learning algorithms; software quality

I. INTRODUCTION

Many of the studies are being done to predict the model behavior for a specific dataset in order to identify the defective modules and faulty modules in a class. These studies are carried out so that the researchers can identify the faults in a class so as to make the software of best quality to their extent. Many of the researchers have given their unique ways to improve the quality of the software. In all these studies, all of them have used classifiers to identify the faulty modules with the help of software metrics in their respective studies. Some of the researchers used NASA datasets [9] while some of them chose other open source datasets. The main aim of this paper is to give an overview to some of the papers and summarize their work so that the researchers may find it easy for them to identify and evaluate the current work that has been in trend today. The study includes the survey of year 2015 and 2016. This will help the researchers to channelize their work in a specific direction.

Below here we are going to discuss some of the terminology so that the researcher or the readers find it easy to understand it in a better way:

A. *Defect prediction*: Any kind of flaw or imperfection in a software product or process is known as defects, and the defect prediction is just a technique by which one can identify the defects.

B. *Software Metrics*: It is a quantitative estimation of a degree to which a system or a process has a given trait or a property. The main aim of this metrics is to identify and control those parameters which affect the software development.

C. *Classifier*: These are the models which help the researchers to identify whether the software has a defect or not. These are the algorithms which take data as an input and process the data and give the output in the form of number of defects. The classifiers are also termed as learners.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

II. RELATED WORK

In [1], the author improved the performance of the web applications by using metrics generated from the CSS source code. The datasets are generated from the open source web applications. And then, defect prediction is performed using 3 different machine learning algorithms.
In [2], the author proposed a methodology on the basis of log transformation to improve the quality of metrics. The effect of log transformation was analyzed on software metrics to identify fault-prone areas on multi-releases of 11 products (41 releases). The outcomes demonstrated that the log transformation can be utilized to determine threshold values for all metrics. The outcomes can then used to lead fault-proneness classification on the basis of threshold values and look at against the outcomes without change.
In [3], author performed a methodical survey studies about from January 1991 to October 2013 that utilizes machine learning algorithms for software fault prediction. We survey the execution capability of various machine learning algorithms in existing research for fault prediction. Author likewise thought about the execution of the machine learning algorithms with the statistical and other machine learning algorithms. At last, the strengths and weaknesses of machine learning techniques are also summarized.
In [4], the author analyses and compares the statistical and machine learning methods for fault prediction. With a specific end goal to analyze and look at the models predicted utilizing the regression and the machine learning strategies they have utilized two openly accessible datasets AR1 and AR6. They have analyzed the predictive capacity of the models utilizing the Area Under the Curve (AUC).
In [5], the author has twofold objectives: (1) investigation of the exertion expected to settle software shortcomings and influenced by different components or factors and (2) forecast of the level of implementation exertion for settling the fault on the basis of information given in software change demands. The work is based on data related to 1200 failures, extracted from NASA system. The investigation included descriptive and inferential insights.
In [6], the author aims to evaluate a relationship between software metrics and software quality in an open-source software projects, and to compare these metrics to other metrics of source code and process. The author performed inferential statistics on open source software projects in order to analyze code ownership metrics and their relationship with software quality.

III. LITERATURE SURVEY

Ruchika Malhotra [7] has demonstrated an empirical framework for android software's defect prediction utilizing various machine learning methods. The paper uses object oriented metrics for predicting defective classes using machine learning technologies. The datasets were obtained from Google Git repository [8] which are the seven application packages of android software, viz. Contact, MMS, Bluetooth, email, calendar, gallery2 and telephony as shown in Table [1]. The results are validated using 10-fold cross validation and inter-release validation methods. Later, statistical test and post-hoc analysis was performed to evaluate the reliability and significance of the results. Calculation of object oriented metrics was done using CKJM tool [9]. The performance measures used for defect prediction on 7 application packages using machine learning technologies are Sensitivity, specificity, area under curve (AUC) utilizing Receiver operating character (ROC). Then the defect prediction performance was compared between 10-fold and inter-release validation where, inter-release validation gave the better result than 10-fold validation. After that, the best and the worst machine learning technique was found using Friedman test which gave ranking to all the machine learning techniques on the basis of Average AUC values. At last, pair of machine learning techniques was found out which were statistically different from each other by performing post-hoc analysis using nemenyi test.
In conclusion, author concludes that the defect prediction, models are efficient and effective, and the author proposes to perform the same work using meta heuristic techniques and hybridized algorithms.
Arvinder Kaur and Inderpreet Kaur [10] showed the empirical evaluation of the six classifier algorithms for fault prediction in an open-source software projects. The aim was to compare the classification models for the fault prediction on an open source projects on the basis of Accuracy, Sensitivity, Specificity, Precision, F-measure, G-mean and J-coefficient. Graphical methods were also used like ROC, Precision-Recall Curve, Cost Curve and Lift charts. It was also aimed to compare the results of an open source software projects with the industrial projects. Dataset was collected from the open-source website SOURCEFORGE [11] as shown in Table [2]. To calculate the values of object oriented metrics CKJM Extended Tool was used. Bugs are collected from the source code using openly available Bug

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Tracker from SOURCEFORGE. Classification algorithms used in this study are Naïve Bayes, Logistic Regression, IB1, J48, Bagging and Random Forest. On the conclusion it has been found that Random Forest gave the best results followed by Bagging. Naïve Bayes gave the least performance. It has also been found that studies on industrial datasets and these datasets produced almost similar results. And hence, the study would help to increase the statistical validity for future studies.

Read Shatnawi [12] has filtered out the less complex parts of the software and the remaining part was used for further analysis. The paper aims to reduce the implementation size and the design size, for which modules at 10%, 20%, and 30 % of less complex LOC and less complex NPM respectively was filtered out of trained and tested datasets The remaining modules are build four classifiers viz. Naïve Bayes, Logistic Regression, k-nearest neighbor, and C4.5 decision tree. Dataset is shown in Table [3]. Fault fixes and software metric values were gathered from the archives of the software projects [13]. The data is composed of source code, change history, and defects data. Datasets were formed by utilizing three tools, infusion tool for converting Java code to FAMIX models; Moose tool for reading FAMIX models and for calculating the number of source code metrics; and Churrasco tool for extracting bug data and for linking the modules [13]. Faulty data was gathered by studying the code subversions or CVS. The bug fixes are mapped to the affected parts of the system. ROC curves were used to analyze the result of the classifiers.
In conclusion, the author says that the smaller modules can be removed without degrading the performance of the classifiers. The software engineers can spend less effort and can direct their efforts to the most vital parts of software. In future, the researcher plan to expand this study to more diverse datasets.

Ahmed H. Yousef [14] has utilized the data mining way to present the attributes that foresee the faulty state of the modules. The paper has collected the data from the NASA REPOSITORIES [15] viz. CM1, JM1, KC1, KC2 and PC1. It uses 4 data mining algorithms viz. Naïve Bayes, Neural Network, Decision Tree and Association Rules. On running these algorithms using the above datasets, the paper gives top-20 attributes in ascending order of their defective state, of each algorithm separately. Then the performance of each classifier algorithm was compared on the basis of accuracy, precision, recall and f-measure. At last, the combined effect of all 4 classifiers was evaluated using Weighted Voting algorithm.
In conclusion, weighted voting algorithm produced much better results as compared to individual algorithms. Naïve Bayes gave the best result individually. In future, the author wants researchers to replicate the work with new software projects and validate the results.

Huanjing Wang, Taghi M. Khoshgoftaar and Amri Napolitano [16] utilized thirty wrapper-based feature selection methodologies to evacuate redundant software metrics utilized for better defect predictions. In this paper, these thirty wrappers depended on the search methods utilized (Best First or Greedy Stepwise), learners (NB, SVM, and LR), and performance measurements utilized (Overall Accuracy, Area Under ROC Curve, Area Under the PR Curve, Best Geometric Mean, and Best Arithmetic Mean) in the evaluation of defect prediction. The outcomes show that Best Arithmetic Mean gave the best result inside the wrapper. Naïve Bayes performed significantly better than Logistic Regression and Support Vector Machine as a wrapper learner on slightly and less imbalanced datasets. On the other hand the models built with full datasets, the performances of models can be enhanced when metric subsets are chosen through a wrapper subset selector. The metric values and faulty data for this study were collected from a real world software project viz. the Eclipse project [17]. We took three releases of the Eclipse system viz. 2.0, 2.1, and 3.0.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Data set name	Version	Total classes	Total LOC	Defective #	Defective %
Gallery2	4.0.2	305	18,853	24	7.86
	4.0.4	310	19,290	69	22.25
	4.1.2	330	20,446	170	51.51
	4.2.2	374	28,223	93	24.86
	4.3.1	647	50,168	130	20.09
Contact	2.3.7	85	7788	29	34.11
	4.0.2	325	22,722	19	5.84
	4.0.4	331	22,834	107	32.32
	4.1.2	357	24,644	35	9.8
	4.2.2	375	25,860	14	3.73
	4.3.1	210	14,807	98	46.66
Email	2.3.2	385	25,760	21	5.45
	2.3.7	394	26,839	41	10.41
	4.0.2	469	33,730	17	3.62
	4.0.4	624	43,147	274	43.91
	4.1.2	475	34,255	61	12.84
	4.2.2	472	34,025	7	1.48
	4.3.1	472	34,037	77	16.31
MMS	2.3.7	195	13,157	54	27.69
	4.0.2	201	13,538	11	5.47
	4.0.4	206	13,804	68	33.01
	4.1.2	223	14,759	42	18.83
	4.2.2	225	14,932	12	5.33
	4.3.1	224	14,915	23	10.27
Calendar	4.0.2	77	8042	22	28.57
	4.0.4	78	8216	45	57.69
	4.1.2	86	9285	12	13.95
	4.2.2	88	9465	41	46.59
Bluetooth	4.1.2	39	2517	15	38.46
	4.2.2	63	6246	10	15.87
	4.3.1	72	7550	13	18.05
Telephony	4.2.2	249	30,325	137	55.02
	4.3.1	224	28,331	154	68.75

Table 1: Summary of various releases of Android Software over application packages

Author transformed the original data by (1) removing all non-numeric attributes, including the package names (2) changing the after-release defects attribute into binary form viz. fault-prone (fp) and non fault-prone (nfp). Participation in each class was decided by after-release defects threshold t, which isolates fault-prone from non fault-prone packages by classifying packages with t or more after-release defects as fp and remaining as nfp. They used t (10; 5; 3) for release 2.0 and 3.0, while t (5; 4; 2) for release 2.1. Table [4] presents details about the datasets used in this study.

On conclusion, the comparison is shown between the performances of two search methods used within wrapper. GS performed better than BF in most cases. In terms of wrapper metrics performance measures, BAM is the best wrapper metric on average. Among the three learners used inside wrapper-based feature selection algorithm, NB was the best learner. Among the three external learners, the performance of SVM was the best, while NB performed worst.

In future, the experiment may include using more learners, other feature subset method and some more metrics.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Project	Total classes	Faulty classes	Faulty %	Description
PMD	104	60	57.7	Programming Mistake Detector PMD
FIND BUGS	226	89	39.4	It looks for bugs in Java code
EMMA	104	61	58.65	Used for measuring and reporting java code coverage
TROVE	250	10	4	Provides a free implementation for java collection API
Dr JAVA	405	81	20	It is a programming environment for java

Table 2: Project Descriptions

Dataset	#Modules	%NFP	%FP
Eclipse JDT Core www.eclipse.org/jdt/core	997	14%	86%
Equinox Framework www.eclipse.org/equinox/	324	40%	60%
Mylyn www.eclipse.org/mylyn/	1862	13%	87%
Eclipse PDE UI www.eclipse.org/pde/pde-ui/	1497	21%	79%
Apache Lucene www.lucene.apache.org	691	9%	91%

Table 3: The fault distributions of all systems

Ruchika Malhotra and Ankita Jain Bansal [18] built the predictive models to identify those parts of the software that have high probability of fault occurance in it. They have considered the impact of thresholds of software metrics on fault proneness for building the predictive models. In this research, author has utilized a statistical model gotten from logistic regression to compute the threshold values of software metrics. By utilizing the thresholds, one can partition the classes into two levels of hazard – generally low and high hazard. They have additionally demonstrated threshold impacts at different hazard levels and approved the utilization of these thresholds on an open domain. The most essential point of this paper is to think about the impact of thresholds of software metrics on fault proneness. Once the threshold values are calculated, one can compare them with the metric values for all the classes. The more the number of metrics in a class with values above their corresponding threshold values, the higher is the chances of faults in that class. To calculate the threshold values, they have used a methodology proposed in [19] based on logistic regression. The results showed that thresholds are quite effective and useful to indicate the high-risk classes. The empirical validation is done on software from NASA viz. KC1 [20] and other two openly available Promise datasets, Apache Ivy and JEdit. To demonstrate the effectiveness, inter-project validation has also been carried out on three open source datasets, Apache Ant and Apache Tomcat and Sakura. Inter-project validation means using different datasets for testing and training. Using training set different from test set may give better prediction results [21].

Project	Data	#Metrics	#Modules	%fp	%nfp
E1	E2.0-10	208	377	6.1	93.9
	E2.1-5	208	434	7.83	92.17
	E3.0-10	208	661	6.2	93.8
E2	E2.0-5	208	377	13.79	86.21
	E2.1-4	208	434	11.52	88.48
	E3.0-5	208	661	14.83	85.17
E3	E2.0-3	208	377	26.79	73.21
	E2.1-2	208	434	28.8	71.2
	E3.0-3	208	661	23.75	76.25

Table 4: Software Datasets Characteristics

Authors have validated the threshold values of Ivy on Ant and Tomcat and the threshold values of JEdit on Sakura dataset. In other words, authors have performed interproject validation on the projects of similar nature. Ivy, Ant and Tomcat are apache software, all implemented in Java language, whereas JEdit and Sakura are test editors. The study has used various machine learning methods (BayesNet, NaïveBayes, random forest, support vector classifier and multilayer perceptron) in order to predict faulty classes. The

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

validation technique used for this purpose is K-cross validation, where the value of K is set to 10, and the performance is evaluated using receiver operating characteristics (ROC) curve [22]. Author have used various data sets from Promise data repository viz. KC1 class level, Ivy, Tomcat, Ant, and JEdit which is collected and validated by the Metrics Data Program (MDP 2006) [23]. Besides these datasets, Sakura editor is also used as shown in Table [5]. Sakura is available under the GNU General Public License 2.0, whereas Sakura is also available under Shareware license. Authors calculated the threshold values of metrics using logistic regression. Thresholds are calculated for KC1, Ivy and JEdit datasets at different risk levels between 0.01to 0.15, and results are compared to conclude that one of the risk level is the best one. Using the threshold values obtained at the best risk level, models are validated using various machine learning methods, viz. BayesNet, NaïveBayes, random forest, support vector classifier and multilayer perceptron. Aside from this, interproject validation is additionally completed utilizing three open source datasets, Apache Ant, Apache Tomcat and Sakura.

Dataset	Version	Instances	Faulty Instances
KC1	Class-Level	145	60
Ivy	2	352	40
Ant	1.7	745	166
Tomcat	6	858	77
JEdit	4.3	492	10
Sakura	2.0.2.0	80	47

Table 5: Brief details of open source datasets used

In conclusion: (1) Univariate logistic regression is conducted for KC1, Ivy and JEdit datasets to find the threshold value. (2) For KC1, the best risk level (i.e. positive values) is 0.15. For Ivy, the threshold values at the risk level 0.07 and higher are within the observation range of all the metrics. But the author selected the lowest value, viz. 0.07. For JEdit, the lowest risk level is 0.02. (3) Results of validation on KC1 concluded that support vector classifier is to be the best machine learning method. The results of support vector classifier showed high g-mean (69.59) and AUC (0.693). When the data is converted to binary using threshold values of KC1, the model showed higher g-mean (72.08) and AUC (0.739). Results of validation on Ivy dataset concluded that the binary models using random forest and multilayer perceptron have performed better than non-binary model. (4) Interproject validation carried was out of the threshold values of Ivy on Ant and Tomcat, whereas threshold values of JEdit on Sakura concluded that the proposed threshold methodology can be used on the similar type of projects.

IV. COMPARISON TABLE

S No.	Author's Name	Published in	Dataset Used	Objective/ Year of Publication
1	Ruchika Malhotra	Elsevier-Science Direct	Table 1	Defect prediction in android software (2016)
2	Arvinder Kaur, Inderpreet Kaur	Elsevier-Science Direct	Table 2	Fault Prediction in open source software projects (2016)
3	Raed Shatnawi	Springer	Table 3	Filtering of less complex parts of software (2016)
4	Ahmed H. Yousef	Elsevier-Science Direct	CM1, JM1, KC1, KC2, PC1	data mining approach for defect prediction (2015)
5	Huanjing Wang, Taghi M. Khoshgoftaar, Amri Napolitano	World Scientific	Table 4	Wrapper based feature subset selection to remove more redundant software metrics used for building defect predictors (2015)
6	Ruchika Malhotra, Ankita Jain Bansal	Expert Systems	Table 5	build predictive models to identify parts of software that have high probability of occurrence of faults (2015)

Table 6: Tabular Representation of all Papers Considered in this survey

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

V. CONCLUSION AND FUTURE WORK

This survey was done so that the researchers may find out the current researches that have been done and the researches that can be done in the future. This survey was based on improving the software quality using software metrics that are applied on various machine learning methods. In all the papers above, the authors have done different researches using different datasets and different machine learning algorithms. All aimed at improving the software quality by removing the defects or faults in the software. This gives the brief to the researchers about the current trend in the particular domain and also helps the researchers on how to channelize their work in the future so that the readers and the researchers may find some more innovation in this field.

REFERENCES

1. M. Serdar Bicer, Banu Diri, "Defect Prediction for Cascading Style Sheets", Elsevier- Science Direct, Applied Soft Computing, Volume 49, Pages 1078-1084, December 2016.
2. Raed Shatnawi, "Deriving metrics thresholds using log transformation", Journal of Software: Evolution and Process, DOI: 10.1002/smr.1702, Volume 27, Pages 95–113, January 6, 2015.
3. Ruchika Malhotra, "A Systematic Review of Machine Learning Techniques for Software Fault Prediction", Elsevier- Science Direct, Applied Soft Computing, Volume 27, Pages 504-518, February 2015.
4. Ruchika Malhotra, "Comparative analysis of statistical and machine learning methods forpredicting faulty modules", Elsevier- Science Direct, Applied Soft Computing, Volume 21, Pages 286-297, August 2014.
5. Maggie Hamill , Katerina Goseva-Popstojanova, "Analyzing and predicting effort associated with finding and fixing software faults", Elsevier- Science Direct, Information and Software Technology, Volume 87, Pages 1–18, July 2017.
6. Matthieu Foucault, Cédric Teyton, David Lo, Xavier Blanc, Jean-Rémy Falleri, "On the usefulness of ownership metrics in open-source software projects", Elsevier- Science Direct, Information and Software Technology, Volume 64, Pages 102-112, August 2015.
7. Ruchika Malhotra, "An empirical framework for defect prediction using machine learning techniques with Android software", Elsevier- Science Direct, Applied Soft Computing Volume 49, Pages 1034-1050, December 2016.
8. URL: https://android.googlesource.com
9. URL: http://gromit.iiar.pwr.wroc.pl/pinf/ckjm/metric.html
10. Arvinder Kaur, Inderpreet Kaur, "An empirical evaluation of classification algorithms for fault prediction in open source projects", Production and Hosted by Elsevier- Science Direct, Journal of King Saud University- Computer and Information Sciences, 23 April 2016.
11. URL: https://sourceforge.net/
12. Raed Shatnawi, "Identifying and eliminating less complex instances from software fault data", Springer, International Journal of System Assurance Engineering and Management, 24 December 2016.
13. D'Ambros M, Lanza M, Robbes R (2010), "An extensive comparison of bug prediction approaches", 7th IEEE working conference on mining software repositories, In: Proceedings of MSR 2010, pages 31–41, 2010**.**
14. Ahmed H. Yousef, "Extracting software static defect models using data mining", Production and Hosted by Elsevier- Science Direct, Ain Shams University- Ain Shams Engineering Journal, Volume 6, pages 133–144, 2015.
15. **URL:** http://promise.site.uottawa.ca/SERepository/datasets-page.html
16. Huanjing Wang, Taghi M. Khoshgoftaar, Amri Napolitano, "An Empirical Investigation on Wrapper-Based Feature Selection for Predicting Software Quality", World Scientific Publishing Company , International Journal of Software Engineering and Knowledge Engineering, DOI: 10.1142/S0218194015400057, Vol. 25 No. 1, pages 93–114, 2015.
17. T. Zimmermann, R. Premraj and A. Zeller, "Predicting defects for eclipse", Proceedings of the 29th International **c**onference on Software Engineering Workshops, Washington, DC, USA, pages 76–85, 2007.
18. Ruchika Malhotra, Ankita Jain Bansal, "Fault prediction considering threshold effects of object oriented metrics", Expert Systems Article, Wiley Publishing Ltd, Vol. 32, No. 2, April 2015.
19. Bender, R., "Quantitative risk assessment in epidemiological studies investigating threshold effects", Biometrical Journal, Volume 41, Pages 305–319, 1999.
20. NASA. (2000) Metrics data repository, http://www.mdp.ivv.nasa.gov/
21. He, Z., F. Shu, Y. Yang, M. Li and Q. Wang, "An investigation on the feasibility of cross-project defect prediction", Automated Software Engineering, DOI: 10.1007/s10515-011-0090-3, Volume 19, Pages 167–199, 2012.
22. El Emam, K.S. Benlarbi, N. Goel and S. Rai, "A validation of object-oriented metrics", Technical report ERB-1063, NRC, 1999.
23. URL: http://sarpresults.ivv.nasa.gov/ViewResearch/107.jsp)

BIOGRAPHY

Prabujeet Kaur is an M.tech Student in the Computer Science and Enginnering Department, Kamla Nehru Institute of Technology, Sultanpur, UP, India. She received a B.Tech degree in 2012 from Shri Ram Murti Smarak Women's College of Engineering and Technology, UP Technical University, Bareilly, UP, India. Her research interests are Cryptography and Network Security, Software Engineering, and Data Mining.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Dharmendra Lal Gupta is currently working as an Associate Professor in the Department of Computer Science & Engineering, Kamla Nehru Institute of Technology, Sultanpur, UP, India. He received B.Tech. degree in 1999 from Kamla Nehru Institute of Technology, Sultanpur, UP, in Computer Science & Engineering, M.Tech. Hon's degree in 2003 in Digital Electronics and Systems from Kamla Nehru Institute of Technology, Sultanpur, UP, India. He has been a member of IEEE Computer Society. He has published about 26 papers in International/National Journals/workshops/conferences and seminars. His research interests are Software Quality Engineering, Software Engineering, Cryptography & Network Security.