IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-10, Oct 2018 E-ISSN: 2347-2693

# Software Bug Prediction and Handling Using Machine Learning Techniques: A Review

**Tamanna[1*], Om Prakash Sangwan[2]**

[1,2]Dept. of computer science and engineering, Guru JamheshwarUniversity of Science and Technology Hisar,India,

[*]*Corresponding Author: tamannasharma100@gmail.com*

**Available online at: www.ijcseonline.org**

Accepted: 19/Oct/2018, Published: 31/Oct/2018

***Abstract*—** It is Impossible to build a software which is completely tested or bug free. Manual bug fixing is very time taking, costly and clumsy task. To automate the process of software bug fixing various machine learning techniques are employed. Software bug prediction is implemented before testing phase of software development life cycle model while bug handling is a post testing phase arises after the failure of test cases. Software bug handling deals with the phases of software bug life cycle model. Bug reports are one of the most important software artifacts for handling of bugs. In recent years, due to release of thousands of open source software, large amount of repositories (like bug repositories) are available for software analytics. Analytics help software practitioners in taking decisions with logic instead of intuitions which make it more accurate and practical. Prediction and Handling of software bugs uses this analytics in automation with the help of machine learning techniques. In this paper we focused on predictive capability of different machine learning techniques in association with software bug prediction and handling. Findings and previous work is summarized with the help of tables (in association with attributes) and diagrams (in mapping with software bug life cycle model).

***Keywords*—** Software Bug, Computational Intelligence, Analytics, Naïve Bayes (NB), Support Vector Machine (SVM)

## I. INTRODUCTION

Failures like 2YK , ARIANE 5 and recent failure of payment software which left 6.5 million customers of RBS(Royal Bank of Scotland) without payment.“Prevention is better than cure” and software evolution supports this saying because any Malware in software can cause fatal consequences which can involve human life as well. Bug report shown in fig.1 describe all the attributes of a standard bug report. Existing research in bug handling took the help of various intelligent techniques such as text mining for bug report analysis, automatic bug triaging, bug fix time prediction and duplicate bug prediction. Track record of various defects, bugs, or issues in software evolution is recorded with the help of software known as bug tacking system.

Software Repositories contains data in structured (like SRS), semi structured (like comments in source code) and unstructured (like developer to client conversation). Different mining techniques are employed (like text mining, information retrieval, classification and clustering etc.) for extraction of useful information. This historical information is used for training of different predictive algorithms.

Bug handling can be very well understood with the help of bug life cycle model (BGCM). BGCM constitutes from submission of issue – unconfirmed to new – new to resolved – resolved to reopen/closed. In this study we map our findings with respect to phases of BGCM.

Rest of the paper is organized in various sections, Section 1 is about introduction of various techniques. Section 2 describes open issues in software bug handling from unconfirmed to closed mapped according to different phases of software bug life cycle model. Section 3 summarizes previous studies with specific attributes in tabular form and section 4 is about conclusion and future scope of machine learning techniques in prediction and handling of software bugs.

| **Category** | Bug ID | | | | Bug Overview | | | Bug Details | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Label** | ID No. | Name | Reporter | Submit Date | Summary | URL | Screenshot | Actual Result | Expected Result | Description | Severity |

Fig. 1. Format of a bug report [9]

## II. OPEN ISSUES IN SOFTWARE BUG HANDLING

*A. Unconfirmed to New:* When a report is submitted it is not confirmed whether it is a bug or an issue. Bug report analysis is a critical task because wrong analysis resulted in to false predictions. Whenever an issue arises issue report is

 

generated but sometimes due to insufficient knowledge of reporter it is misclassified as bugs. Empirical study carried out on Mozilla, JBOSS and Eclipse with the help of decision trees and naïve bayes shows 77% of issues are classified correctly [7]. Data collected from repositories are not always qualitatively fine. Survey is carried out on both open source and closed source projects, shows that differences between projects in terms of link rate (between bugs and commits) resulted in lack of quality[9]. Descriptive model is built on the basis of statistical analysis of surface features shows that quality increase and cost decrease in bug prediction [13]. Data gathered from open repositories are not noise free. Therefore, removal of redundant data from these large bug repositories is very clumsy task. In table1.open issues from various research papers are summarized focusing on quality of bug report. Study shows that integration of different repositories (versioning system and bug system) resulted in to missing traceability links because of heterogeneity between the repositories[24].Missing links between heterogeneous repositories are not able to project complete software evolution

An approach was proposed in which populate one repositories with the help of another repositories and merging point shows complete evolution of software with improved reasoning capabilities[34].

Bugs are uniformly distributed in the database. Analytical study is accomplished on two types of bugs surprise bugs and breakage bugs with imbalanced data and text classification algorithms and shows that bugs are not classified uniformly it may be possible

that only a small portion of code may contain large number of bugs [25].Empirical study on bug reports shows that it is helpful in knowing the expertise of developers [37]. Study of 72482 bug reports of UBUNTU (9 releases) is experimented for the prediction of corrective maintenance. Correction time of bug reports is predicted with the help of linear model [26].

*B. New to Resolved:* After the confirmation of bug next step is resolution, bugs are assigned to the developer for resolution known as Triaging. Due to large number of variance in bugs and developers manual assignment becomes tough and time taking. Extended version of Latent Dirichlet topic model named multi feature topic model in association with Rapid miner is employed for mapping of bug reports to topic space [17].

Table 1 summarized all the issues related with the dataset taken from bug reports.

| ISSUE ADDRESSED | STUDY |
|---|---|
| In-efficient use of datasets | • Structured information taken in to consideration (BLIA).[1]<br>• Fellegi-Sunter Model is used for integration of databases shows effective results in recovering of traceability links.[2]<br>• Empirically tested differences in quality of datasets cause differences in results .[9]<br>• Tested unfairness, imbalanceness of datasets with the help of prediction techniques .[5] |
| Misclassification of bugs | • Empirically observed misclassification impact the quality of prediction.<br>• Study shows 39% of files actually never have bugs.[6] |
| Lack of coordination affects quality | • Empirical study verifies that rich bug histories fill the gap between social, organizational and technical aspects .[8] |
| Non standardization of bug reports | • Recommender system named CUEZILLA was proposed for measuring quality and recommends elements for the improvement of bug reports.[14] |
| How to mine Bug Summaries qualitatively | • Text mining techniques are employed to find frequent patterns of bugs from the bug summaries.[10]<br>• FRASR(framework for analyzing software repositories) in association with process mining.[11]<br>• Textual coherence of user comments also shows promising results in bug prediction.[12] |

 

All bugs are not same in nature and divided in terms of criticality. According to "Bugzilla" it is labeled as blocker, critical, major, minor and trivial. BM-25 based document similarity function is proposed and based on the history of bug report severity database label of new report is predicted [19].Association mining is used with a priori algorithm on summary terms and predicts the developer for bug fixing. Experiments on various open source projects shows that approach based on association mining reduces the development effort and time [33].Automatic assignment of bug report to developer reduces cost and time of fixing of bugs.

*C. Resolved to Closed:* After the reporting of bugs it is passed to triage team, assigned to someone for fixing and then they are verified and after that closed but sometime problem arises (reopening of bugs) which results in increase of software maintenance cost. resolved to closed is three tier process. *ReopenPredictor* is proposed based on extraction of textual features and help in prediction reopened bugs .

## III. SUMMARY OF INTELLIGENT TECHNIQUES INCORPORATING WITH SOFTWARE BUG PREDICTION AND HANDLING

Papers are summarized with respect to machine learning technique employed and its key points like dataset, features, efficacy measures and experimental results. These findings are divided in to two parts first one deals with software bug prediction and other is about software bug handling. Tabular representation in table 2 of software bug prediction techniques focused firstly on papers which tries to improve the prediction capability of software bugs (by tuning of learning parameters or by feature selection, instance selection etc.). While handling deals with predictive techniques which make software bug handling automated, fast and anticipated. These are like bug severity prediction, developer prioritization, bug triaging and duplicate bug removal etc.

Table 2. Summarizes different Techniques used for Software Bug Prediction and Handling

| Reference | Dataset | Techniques Used | Feature Employed | Efficacy Measures | Experimental results |
|---|---|---|---|---|---|
| [25] | ANT 1.7 dataset of promise repository | Gradient descent with adaptive learning back propagation (GDA) | _ | Accuracy and performance | Proposed model provide 98.97% accuracy |
| [27] | Predict defects between six open source projects | GA + Regression tree and generalized linear model maximize the ratio between the number of defects found and effort used | _ | Recall, Precision, AUC | Regression Models Using GA significantly outperform their counterparts<br>Future approach is employed on Bayesian or NN to other software metrics |
| [28] | Study perform on 26 open source systems | Build a hybrid prediction model in conjunction with code changes information. | Eleven predictors exploited by five competitive techniques | _ | Outperform four baseline techniques based on process metrics and future involves investigation of factors causing scattering to developers. |
| [29] | Six large open source projects BUGZILLA, COULMBIA, JDT platforms Mozilla | Ensemble learning approaches bagging and stacking together with random forests. | Containing a total of 137,417 changes | Cost effective and F1 Score | TLEL outperforms<br>three baselines across the six datasets and will optimize the parameters of TLEL |
| [30] | Baseline extensive dataset is composed | Five systems are used | Source code metrics, past defects | _ | Concluded that prediction technique based on single metric do not work consistently |
| [31] | Five open source java systems | Feature selection is done automatically with regularization method on linear and poison regression | Software metrics are taken as features / independent variables | Root mean square error up to 50% | Regularization methods reduce the prediction error of regression and improve stability<br>39 studies out of 64 doesn't apply feature selection method |
| [20] | _ | Applied Naïve Bayes and SVM classifier for prediction of bugs in conjuction with change information | Changes are taken as features | _ | Future where software engineers commit it as a buggy or clean |
| [32] | Eclipse dataset | Ginni coefficient is used to quantify the | | AUC and | Lorenz curve is used for empirically |

 

| | | | | | |
|---|---|---|---|---|---|
| | | inequality of a distribution<br>Random forest analyze the relationship between code ownership and bugs in source files using Ginni co-efficient | _ | LORENZ curve | investigation between the code ownership and bugs in source files<br>Future other than eclipse is used for experimentation and source code changes |
| [34] | Five open source java systems | Machine learning adjustable parameters called hyper parameters<br>KNN and SVM | Multi search WEkA tool is employed | _ | Tuning hyper parameters gives at least as accurate models for SVM<br>Plan to extend this study to cover more machine learning models |
| [24] | Seven Datasets of Eclipse Projects Mozilla Product | Summary terms are extracted with text mining technique KNN, SVM and NB are used | Bug summary Attribute | F-measure, Accuracy, Precision and Recall | KNN performs better than the support vector machine.<br>Try to develop cross project bug severity predictions |
| [9] | Open source project, Aspect J, SWT, and Zxing. | Bug localization using integrated analysis (BLIA) is proposed on method level | Texts, Stacks, Traces and comments. | Average Precision, MAP, MRR are used | Determine BLIA's potential for improving the accuracy of bug localization at method level |
| [26] | Mozilla, Gnome and Eclipse | Text mining algorithm | Tf-IDF Modelling | _ | Frequently used terms are used to describe bugs |
| [35] | 1200 failures extracted from the change tracking system of a large NASA mission | Three supervised machine learning and three sampling technique | _ | _ | 83% of total fix implementation effort was associated with only 20% of failures |
| [36] | _ | Use instance selection with feature selection to reduce bug data scale | _ | _ | Which developer should be assigned is based on text categorization results. |
| [37] | Eclipse and Mozilla | Bug triaging, severity identification and reopened bug prediction based on developer prioritization | _ | _ | Empirically investigate the model and proved that developer prioritization positively impact bug triaging |
| [38] | NETBEANS and Eclipse using BUGZILLA bug repositories | Linearly discrimnant analysis (LDA) , NB<br>PCA for feature selection and PSO for instance selection | Bug report instances | Accuracy, precision, Recall and processing time | Feature selection to instance selection gives good results than instance selection to feature selection |

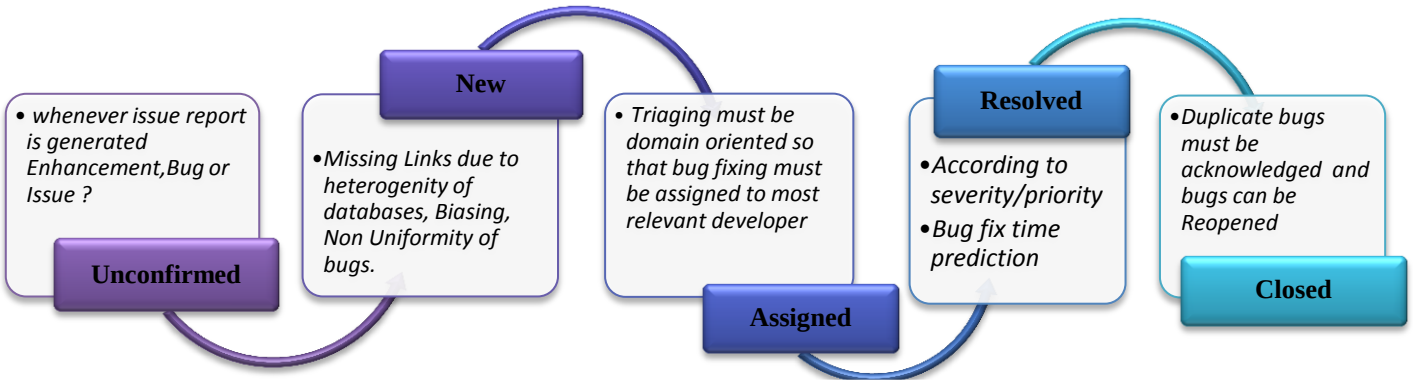


Fig.2 Issues encountered at every stage of software Bug life Cycle

 

## IV. MAJOR FINDINGS AND CONCLUSION

In this paper we presented machine learning techniques employed for prediction and handling of software bugs. Software bug handling used different computational techniques for making it automated, efficient and fast but these techniques are not completely justifiable and lots of efforts are needed to make it more automated. Review of following papers on bug prediction techniques shows that most of the studies neglect parameter tuning, statistical tests which plays a crucial role in accuracy of machine learning algorithms. Other open issues concluded with respect to the phases of SBLC in fig 2 which can be used for future research. Mapping of future research is done with respect to software bug life cycle model for example: when bug report is submitted it is **unconfirmed** whether it is bug or issue and it is still an open area of research. Similarly when bug is

confirmed it is entered in to **new** state and problems like missing links, biasing, heterogeneity of language between bug reports and source code occurs which takes too much time if done manually. Other phases (like assigned, resolved and closed) also have issues which are very time taking and clumsy for software developers to do manually and correct implementation of intelligent techniques make this work fast, efficient and within in budget of software development cost because of efficient utilization of resources.

## REFERENCES


[1]Youm, K.C., Ahn, J. and Lee, E., 2017. Improved bug localization based on code change histories and bug reports. Information and Software Technology, 82, pp.177-192.

[2]Sureka, A., Lal, S. and Agarwal, L., 2011, December. Applying fellegi-sunter (fs) model for traceability link recovery between bug databases and version archives. In Software Engineering Conference (APSEC), 2011 18th Asia Pacific (pp. 146-153). I

[3]Bhattacharya, P. and Neamtiu, I., 2011, May. Bug-fix time prediction models: can we do better?. In Proceedings of the 8th Working Conference on Mining Software Repositories (pp. 207-210). ACM.

[4]Osman, H., Ghafari, M. and Nierstrasz, O., 2017, February. Hyperparameter optimization to improve bug prediction accuracy. In Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), IEEE Workshop on (pp. 33-38). IEEE.

[5] Bird, C., Bachmann, A., Aune, E., Duffy, J., Bernstein, A., Filkov, V. and Devanbu, P., 2009, August. Fair and balanced?: bias in bug-fix datasets. In Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering (pp. 121-130). ACM.

[6] Herzig, K., Just, S. and Zeller, A., 2013, May. It's not a bug, it's a feature: how misclassification impacts bug prediction. In Proceedings of the 2013 international conference on software engineering (pp. 392-401). IEEE Press.

[7] Antoniol, G., Ayari, K., Di Penta, M., Khomh, F. and Guéhéneuc, Y.G., 2008, October. Is it a bug or an enhancement?: a text-based approach to classify change requests. In Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds (p. 23). ACM.

[8] Aranda, J. and Venolia, G., 2009, May. The secret life of bugs: Going past the errors and omissions in software repositories. In Proceedings of the 31st International Conference on Software Engineering (pp. 298-308). IEEE Computer Society.

[9] Bachmann, A. and Bernstein, A., 2009, August. Software process data quality and characteristics: a historical view on open and closed source projects. In Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops (pp. 119-128). ACM.

[10] Jayadev Gyani, Narsimha G. and Kiran Kumar B 2014, May Mining Frequent Patterns From Bug Repositories .International Journal of Advanced Research

[11] Poncin, W., Serebrenik, A. and Van Den Brand, M., 2011, March. Process mining software repositories. In Software maintenance and reengineering (CSMR), 2011 15th european conference on (pp. 5-14). IEEE.

[12] Dit, B., Poshyvanyk, D. and Marcus, A., 2008. Measuring the semantic similarity of comments in bug reports. Proc. of 1st STSM, 8, p.64.

[13] Hooimeijer, P. and Weimer, W., 2007, November. Modeling bug report quality. In Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (pp. 34-43). ACM.

[14] Khomh, F., Dhaliwal, T., Zou, Y. and Adams, B., 2012, June. Do faster releases improve software quality? an empirical case study of Mozilla Firefox. In Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (pp. 179-188). IEEE Press.

[15] Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schroter, A. and Weiss, C., 2010. What makes a good bug report?. IEEE Transactions on Software Engineering, 36(5), pp.618-643.

[16] Bachmann, A. and Bernstein, A., 2010, May. When process data quality affects the number of bugs: Correlations in software engineering datasets. In Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on(pp. 62-71). IEEE.

[17] Xia, X., Lo, D., Ding, Y., Al-Kofahi, J.M., Nguyen, T.N. and Wang, X., 2017. Improving automated bug triaging with specialized topic model. IEEE Transactions on Software Engineering, 43(3), pp.272-297.

[18] Kumar, R. and Gupta, D.L., 2016. Software Bug Prediction System Using Neural Network. European Journal of Advances in Engineering and Technology, 3(7), pp.78-84.

[19] Tian, Y., Lo, D. and Sun, C., 2012, October. Information retrieval based nearest neighbor classification for fine-grained bug severity prediction. In Reverse Engineering (WCRE), 2012 19th Working Conference on (pp. 215-224). IEEE.

[20] Lamkanfi, A., Demeyer, S., Giger, E. and Goethals, B., 2010, May. Predicting the severity of a reported bug. In Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on (pp. 1-10). IEEE.

[21] Xuan, J., Jiang, H., Ren, Z. and Zou, W., 2012, June. Developer prioritization in bug repositories.
In Software Engineering (ICSE), 2012 34th International Conference on(pp. 25-35). IEEE.

[22] Jain, S. and Wilson, S.R., 2016, August. Automated bug assortment system in datasets. In Inventive Computation Technologies (ICICT), International Conference on (Vol. 2, pp. 1-7). IEEE.

[23] D'Ambros, M., Lanza, M. and Robbes, R., 2010, May. An extensive comparison of bug prediction approaches. In Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on (pp. 31-41). IEEE.


 

[24] Ayari, K., Meshkinfam, P., Antoniol, G. and Di Penta, M., 2007, October. Threats on building models from cvs and bugzilla repositories: the mozilla case study. In Proceedings of the 2007 conference of the center for advanced studies on Collaborative research (pp. 215-Yang, X.L., Lo, D., Xia, X., Huang, Q. and Sun, J.L., 2017. High-impact bug report identification with imbalanced learning strategies. Journal of Computer Science and Technology, 32(1), pp.181-198.228). IBM Corp..

[25] Yang, X.L., Lo, D., Xia, X., Huang, Q. and Sun, J.L., 2017. High-impact bug report identification with imbalanced learning strategies. Journal of Computer Science and Technology, 32(1), pp.181-198.

[26] Anbalagan, P. and Vouk, M., 2009, September. On predicting the time taken to correct bug reports in open source projects. In Software Maintenance, 2009. ICSM 2009. IEEE International Conference on (pp. 523-526). IEEE.

[27] Anvik, J., Hiew, L. and Murphy, G.C., 2006, May. Who should fix this bug?. In Proceedings of the 28th international conference on Software engineering (pp. 361-370). ACM.

[28] Puranik, S., Deshpande, P. and Chandrasekaran, K., 2016. A Novel Machine Learning Approach for Bug Prediction. Procedia Computer Science, 93, pp.924-930.

[29] Arudkar, S. and Pimpalkar, A., Design of an Effective Mechanism For Automated Bug Triage System.

[30] Singla, H., Sharma, G. and Sharma, S., 2016. Domain Specific Automated Triaging System for Bug Classification. Indian Journal of Science and Technology, 9(33).

[31] Singh, V.B., Misra, S. and Sharma, M., 2017. Bug Severity Assessment in Cross Project Context and Identifying Training Candidates. Journal of Information & Knowledge Management, 16(01), p.1750005.

[32] Chaturvedi, K.K. and Singh, V.B., 2012. An empirical comparison of machine learning techniques in predicting the bug severity of open and closed source projects. International Journal of Open Source Software and Processes (IJOSSP), 4(2), pp.32

[33] Sharma, M., Kumari, M. and Singh, V.B., 2015, June. Bug assignee prediction using association rule mining. In International Conference on Computational Science and Its Applications (pp. 444-457). Springer, Cham.

[34] Fischer, M., Pinzger, M. and Gall, H., 2003, September. Populating a release history database from version control and bug tracking systems. In Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on (pp. 23-32). IEEE.

[35] Di Nucci, D., Palomba, F., De Rosa, G., Bavota, G., Oliveto, R. and De Lucia, A., 2018. A developer centered bug prediction model. IEEE Transactions on Software Engineering, 44(1), pp.5-24.

[36] Anvik, J. and Murphy, G.C., 2007, May. Determining implementation expertise from bug reports. In Proceedings of the Fourth International Workshop on Mining Software Repositories (p. 2). IEEE Computer Society.

[37] Osman, H., Ghafari, M. and Nierstrasz, O., 2017, February. Automatic feature selection by regularization to improve bug prediction accuracy. In Machine Learning Techniques for Software Quality Evaluation IEEE Workshop on (pp. 27-32).

[38] Shivaji, S., Whitehead, J.E.J., Akella, R. and Kim, S., 2009, November. Reducing features to improve bug prediction. In Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering (pp. 600-604).

[39] Giger, E., Pinzger, M. and Gall, H., 2011, September. Using the gini coefficient for bug prediction in eclipse. In Proceedings of the 12th International Workshop on Principles of Software Evolution (pp. 51-55).

[40] Xia, X., Lo, D., Shihab, E., Wang, X. and Zhou, B., 2015. Automatic, high accuracy prediction of reopened bugs. Automated Software Engineering, 22(1), pp.75-109.

[41] Xia, X., Lo, D., Wang, X., Yang, X., Li, S. and Sun, J., 2013, March. A comparative study of supervised learning algorithms for re-opened bug prediction. In Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on(pp. 331-334).

## Authors Profile

Tamanna is doing PhD in Software Engineering and Soft Computing from Guru Jambheshwar University of Science and Technology, Hisar, Haryana and Master of Technology in Computer Science and Engineering from Banasthali University, Rajasthan. Her area of research is Software Engineering with Machine Learning, Mining Software Repositories, Software Reliability engineering and Automated Software Debugging.

Dr. Om Prakash Sangwan is presently working as Professor, in department of Computer Science & Engineering, Guru Jamheshwar university of Science and Technology, Hisar. He received his PhD in Computer Science & Engineering and Master of Technology (M.Tech) degree in Computer Science & Engineering from Guru Jambheshwar University of Science & Technology, Hisar, Haryana. His area of research is Software Engineering focusing on Planning, Designing, Testing, Metrics and application of Neural Networks, Fuzzy Logic and Neuro-Fuzzy. He has numbers of publications in International / National Journals and Conferences. He is presently working as Professor, Department of Computer Science & Engineering, GJUS&T.

 