Software Fault Prediction Models Using Soft Computing - A Systematic Review

Gurmeet Kaur[1], Dr. Jyoti Pruthi[2], Dr. Parul Gandhi[3]

[1]*Research Scholar CST, MRU, India, grmtkaur02@gmail.com*
[2]*Professor CST, MRU, India, jyoti@mru.edu.in*
[3]*Professor FCA, MRIIRS, India, parul.fca@mriu.edu*

Abstract.

The most dynamic exploration zone in software engineering is software fault prediction because it offers the benefits regarding time complexity, low cost budget, testing effort, and increases the reliability along with the quality of the software if it is applied at the starting phase of traditional and agile based software development life cycle. This study of literature review is conducted in a systematic manner to understand the trends and techniques used for software fault prediction (SFP) problem and synthesis the qualitative results to present technical and methodological information, success and usefulness of SFP model. From this study, it analyzed that Neuro-Fuzzy as a soft computing methodology provides more accurate result for prediction of faults at the initial and later stage of software development, using a variety of metrics for feature selection process, identified datasets and reporting the statistical significance of different accuracy measure parameters, and comparing the performance of existing models with designed model.

Keywords. Software Fault Prediction, Agile Software Development, Soft computing, Software Metrics, Software defects.

1. INTRODUCTION

The Software Quality Assurance and Software Reliability is the key to guarantee the superior quality of software. Both these concepts are attracted all throughout the development of the software and measure[1]. A software bug is a defect, error, failure, or flaw during the execution of a code of software that permits it from acting as unexpected function (e.g., delivering an erroneous outcome). The software defect may be a defect that gives rise to failure of software functionally. This failure may occur due to the existence of one quiet fault in the software. Defective software modules cause software failures, reduce customer satisfaction, increased development, and maintenance costs. A software fault prediction (SFP) is frequently suggested because it incorporates the activities inside the development interaction which assists with anticipating the defective modules or data at initial stages of the development of software. The forecasting of defective modules at the initial stage makes the process of testing simple and quick. With the expansion to the present, it likewise improves the norm of programming.

With the fast progression of the software business, the use of the agile development technique proposed in recent years stress on timely reacting to the changes in prerequisites as describing the insufficiency of the conventional programming development process. Agile development involves different structures or systems, specifically Scrum, XP, Crystal strategies, Lean Software Development, Feature Driven Development and so on. Agile-based process effectively deals with the truth of variation.

Estimating performance and project progress is a motivating field in programming practice. Inside the traditional programming development process, programming measurements are gathered into product metrics and process metrics. To evaluate the complexity of product, TSD measure normally utilizes Cyclomatic Complexity Metric (CCM), Halstead Complexity Metric (HCM), and Lines of Codes (LOC. Defect Density estimates the defect per function point or defect per KLOC and might be a product reliability metric. Defect Removal Efficiency is one among the significant estimations of programming standard. There are various kinds of soft computing modelling based approaches like a Fuzzy Logic System and Neural Network employed for Software Fault Prediction.

In recent years, various researchers try to automate the fault prediction process by designing computer-based models that can perform learning from existing prediction data. The objective of this research is to take correct decisions on the development of the prediction models on basis of finest knowledge and execution of many previous related studies. The outcomes of this research paper explore the existing fault prediction techniques for agile software. This research paper describes the review of literature in different ways by recognizing the basic studies on SFP and the key features of prediction models as follows:

- Prediction techniques: Based on soft computing techniques like Artifical Neural Network, Fuzzy Inference System, Machine learning techniques, and Neuro-fuzzy Hybrid system;
- Software metrics: Either Product and Process Metrics and Agile based metrics;
- Datasets: Public and Private data sets;
- Performance evaluation methods: Either Continuous and Categorical studies;

2. LITERATURE REVIEW

The goal of the systematic review is to understand technical works that describe modelling and identification of metrics to predict the fault for agile software development (ASD) methodologies in the area of prediction methods in soft computing. The research performed in the following four steps:

Step 1: Conduct manual and automatic search to get an initial list of studies. Always try to discard the duplicate studies.

Step 2: To identify the inherent relevant studies on the basis of title and abstract, and reject studies which are not related to the concern topic.

Step 3: Then, shortlist the research paper reviewed on the basis of the introduction, methodology section and conclusion. If reading of the selected research paper is not complete to confirm results, the study of the research paper read in detail.

Step 4: Finally, perform essential investigation using the esteemed standard of achieved and administered list of research paper.

This section describes a systematic method for the reviews of existing research paper on the fault prediction in source code by identifying research issues.

2.1 Research Issues

To explore the models applied to forecast faults in code, the following research questions are described as follows:

RI1. Which soft computing approaches/methods have used to predict fault for traditional and agile development software?

RI2: Which software metrics have been applied in the fault prediction models for traditional and agile development software?

RI3: What is the level of accuracy of fault prediction models?

2.2 Results of Concern Research

RI1. Which soft computing approaches/methods have used to predict fault for traditional as well as agile development software?

As the growing number of faults affects development time, cost and quality of a software package, so software fault prediction is the way of detecting faulty components in units in the prior of the implementation of the software. This study is predicated on systematic review that provides a comprehensive picture in the field of software fault prediction administered by many researchers for agile based software development.

Catal and Diri (2009) [2] described research on types of methodology or techniques used, metrics, and datasets for software fault prediction. The outcomes shows that the use of machine learning computation increased successively after 2005 then PROMISE data store was designed.

Hall (2012) [3] reported the application of feature selection algorithm and the combination of independent variables gives a better outcome in performance.

Begel, (2007) [4] studied based on an experiment performed to find out regarding agile based development, and its implementation in development, executives and testing by individuals, it had been found that 33% of the research respondents make the use of agile strategies. Around 65% of respondents used scrum in their software teams. The test-driven development and pair programming were the least used practices.

Table 1. Technical description of RI1

Ref. no.	**Objective**	**Methodology**	**Data Set/Features**	**Findings**	**Performance measurement tool**
[5]	The fuzzy inference system proposed to calculate the efficiency of metrics in defect predicting for agile software projects.	Neuro-Fuzzy hybrid approach used	From PROMISE repository	The proposed framework gives superior accuracy (specially for large size projects) as evaluated from performance measurement tools	MMRE, BMMRE, RMSE, NRMSE,
[6]	Proposed new variations of WOA as wrapper algorithms to deal with the feature (metric) selection issues in SFP applications.	Roulette wheel, Tournament, Linear position, and random based, stochastic universal sampling.	From the PROMISE archive.	The deep investigation presented that the suggested TBWOA (tournament approach) exceeded the principal WOA.	Average AUC and running time.

[7]	Proposed a system using inter-version and inter-project assessment, to recognize the product defect.	Based on fuzzy logic, the past assignments or adventure variations are taken for preparing sets, and, the present version or exercises are taken as testing sets.	From PROMISE store, even as on PDE and JDT adventures and other more nine open source adventures.	The assessment results concluded that the suggested system showed outstandingly good outcomes for Eclipse-PDE and Eclipse-JDT based projects.	AUC and GM, Root mean square error RMSE.
[8]	Proposed an exact framework to anticipate programming deficiency and assisted the feature selection algorithm for classification issues.	Proposed upgraded binary moth flame Optimization and adaptive synthetic sampling	It makes use of the wrapper feature selection and improves the data set.	ADASYN attempts to beat the imbalanced data issue while BMFO perform as a feature selection.	Average AUC
[9]	Depicted adequately anticipate the product defect proneness in the soft modules with a tendency to lessen programming maintenance cost.	A DDN (deep neural network) with BAPS (Bound particle swarm optimization) dimensional reduction suggested.	From real-world programming projects (NASA and Eclipse) was made use.	The analysis result showed that the BPSO-based dimensionality reduction innovation can enhance the organization structure and acquire better execution.	F-measure, area under the curve (AUC), and probability of detection (pd).

The study describes many techniques and approaches which are soft computing in nature and used for fault prediction in traditional and agile based software, but one of the study from [5] relates the field of research as Software fault prediction in the best manner. The author presented framework using 21 process metrics incorporating different phases in the life cycle of software development using fuzzy inference system to compute the number of faults and applied the back propagation algorithm to train the fuzzy set of rules to improve the accuracy of prediction and also validate the presented model using 29 projects from the PROMISE repository. The author computed diverse performance criteria such as MMRE (0.0539), BMMRE (0.0585), RMSE (18.69), and NRMSE (0.010). The observed and calculated values concluded that the presented model provides better fault prediction capability as comparing to models discussed in their literature.

There are three categories of datasets as Private, Public, and Unknown. Public datasets are available freely from PROMISE and NASA repository. Private Datasets normally belong to private organizations. The unknown datasets are those which are neither public nor private.

RI2: Which software metrics have been applied in the fault prediction models for traditional and agile development software? Metric can be classified into class-level, method/product-level, process-level, and component-level. Halstead (1977) and McCabe (1976) in 1970 proposed Method/Product-level metrics, and they are still in use.

"Product Metrics"

"Defect per Function Point": It can be calculated using Function Point, which helps in Software fault prediction.

"Defect per SKLOC": It can be helpful in Software fault prediction but this metric will be dependent on language used for implementation [10].

"Class-level metrics" It can be applied to object oriented programming only and proposed by Chidamber–Kemerer in 1994 and known as Chidamber & Kemerer (CK) metrics suite is still used by various researchers and software vendors. CK metric suite is perfect for prediction of fault in source code. These metrics are WMC, DIT, CBO, RFC, NOC, and LCOM [11].

"Process Metrics"

Process measurements incorporate the arrangement of measurements, which relies upon the qualities gathered over the life cycle of software development. The classes of process measurements are Code Churn, Requirement Metrics, Change Metrics, and Code Delta.

"Agile Software Development (ASD) based metrics" Changing needs is one among the principle issues that emerge inside the process of software development. To select software metric for Agile-based software, there is need for deep understanding to get the difference between the Agile-based process and Traditional Software Development (TSD), and their measurements methods.

"Task effort": The team produces and evaluates effort based on software professional hours for every task [12].

"Number of stories": The number of stories within the sprint can be basic, medium, and sophisticated on the basis of the story. The measurement is determined as a value in term of count or weight of the complexity of the story [13].

"Story point": Fibonacci format is used to describe the problem size of a story. There might be difficulty associated with uncertainty, efforts, and complexities involved in the process of measurement to forecast the software effort and size needed [13].

"Velocity": It is computed as the sum of all accepted works [13].

For example if for a project completion there are 5 sprints and story points completed from every sprints are 12,34,30,22,and17, then velocity is computed as follows:

$$\text{Velocity} = (12+34+30+22+17)/5 = 115/5 = 23$$

"Work capacity": The work capacity is the number of chores revealed during the run of the sprint, with respect to the finished feature. [13].

"Focus factor": The proportion among velocity and Work capacity, the great worth reaches approximate 80% for a group [13].

$$\text{Focus factor} = \text{Velocity} / \text{Work capacity} \tag{1}$$

"Open defect severity index": The defect evaluated at the completion of the sprint. This metric measures quality of feature for each and every iteration of the sprint [13].

"Defect per iteration": Calculate the entire defect that was presented during the sprint. The defect that was found, and will not be compile within the sprint [13].

"Error density": Number of mistakes detailed by the client after delivery (determined per sprint) [12].

C. Jones [10] (2008) the investigation inspected that there were a couple of difficulties in estimating the performance of the agile based development. On account of using the knowledge of the TSD Process, numerous project manager and developers are favoured to make use of a comparable measurement for the ASD process.

Padmini (2015) [12] the investigation was intended to gather data about the metric used in the agile at present. The investigation distinguished measurements like unit test coverage, ideal delivery, and defect seriousness index, bug correction time, in ASD measures.

Arisholm, E., L.C., Johannessen, Briand, E.B. 2010 [14] made sensible models with high inadequacy probability to recognize parts of a Java based structure. Because of the extended demonstration of object-oriented development in companies, an extended use of object-oriented metrics has been proposed.

Briand L., Ikonomovski S., Wüst J., and Lounis H. (1998) [15] assembled an adequate data and information regarding the utilization of design measures. The author examined the coupling and cohesion estimation information gathered for the 83 framework classes. The author likewise concludes that some of the aspects to be estimated on the selected data sets: method innovation versus aggregate coupling, coupling to application classes versus library classes, and export versus import coupling.

RI3: What is the level of accuracy of fault prediction methods?

The prediction results of SFP model depend on and vary according to dependent variables. To measure the level of accuracy of the SFP model, there are various methods available that can be applied to analyze the performance of the designed SFP model. These measurements focused on predicting whether a given part of software is fault free or not.

Magnitude of Relative Error (MRE): The most commonly used accuracy metrics are the magnitude of relative error. It can be calculated on the basis of either the mean or the median. MRE is an always less than 1[5].

$$\text{MRE} = \frac{(\text{Actual defect} - \text{Predicted defect})}{\text{Actual defect}} \qquad (2)$$

Mean MRE (MMRE): It is the average of magnitude of relative error values N projects. The problem of the Mean MRE is its sensitivity to anomaly.

Balanced MMRE (BMMRE): As MMRE is unbalanced, so for this reason BMMRE is used as [5][16][17]:

$$\text{BMMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\text{ Actual defect} - \text{Predicted defect}|}{\text{min (Actual defect ,Predicted defect)}} \qquad (3)$$

Table 2. Accuracy values of the prediction methods used.

Ref. No.	Prediction Techniques	Accuracy Model	Accuracy Value
[6]	Machine learning	Accuracy	0.982
[5]	Fuzzy Inference System	MMRE	0.0539
		BMMRE	0.0585
		RMSE	18.69
[7]	Neuro- Fuzzy System	RMSE	0.1266
		GM	0.8128
		Accuracy	84.4941
[16]	Artifical Neural Network	MMRE	0.0247
		BMMRE	0.0244
[17]	Fuzzy Inference System Artifical Neural Network	MMRE	0.36343
		BMMRE	0.36534
		MMRE	0.015397
		BMMRE	0.007258

RMSE (Root MSE): The RMSE is computed as square root of the mean of squared difference between actual and observed/predicted defect values [5][7]. It is evaluated as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}}(\text{Actual defect } - \text{ Predicted defect})^2 \qquad (4)$$

Where, n denotes the extent of the actual dataset. It gives the quadratic based value to compute the average measure of the error.

Accuracy: It is applied to compute and equate the precision of prediction models, and calculated as the ratio of number of fault free predictions executed by the total number of prediction executed [7].

Accuracy = (True Positive +True Negative) / Total number of instances executed (5)

Where, True Positive –the number of fault free cases executed by the certain class of software and True Negative - the number of cases rejected by the certain class of project.

G-Mean: For imbalanced datasets, G-Mean is applied to compute the efficacy of the prediction method [7]. It is calculated as:

$$\text{G-Mean} = \sqrt{Precision * Recall} \qquad (6)$$

$$\text{Where, Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad \text{and}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

3. RESULT AND DISCUSSION

Since the predictability of fault is the essential step for development of any software, and this is also required for projects based on agile methodologies. The literature review study explore that there is very small quantity of efforts has been done in software fault prediction for agile-based software. Applying soft computing techniques for fault prediction in agile software development may result better performance in concern of precision. The SFP model affects in various areas of software development which in turn offer benefits in concern of reliability, quality, time and cost of completion.

4. CONCLUSION

The present research paper conclude that Neuro-Fuzzy (Hybrid) as a soft computing methodology provides more accurate result for prediction of faults, but it requires a real and huge data set to implement a model for agile software project from industry. There is a variety of metrics (Product, Process, and Agile based) that are used for feature selection process. The metric selection would be depending on the size or structure, and number of defects, effort on inspection activities, and experience of team members. This research paper also identified and reported the statistical significance of different accuracy measure parameters. This study also presents the vision of fault prediction field of software engineering by exploring the latest view of literature and a distinctive feature of fault prediction.

5. REFERENCES

[1] Parvinder S. Sandhu, Sunil Khullar, Satpreet Singh, Simranjit K. Bains, ManpreetKaur, Gurvinder Singh, ‘A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm’, World Academy of Science, Engineering and Technology Vol. 72, pp. 648-653, 2010.

[2] Catal C, Diri B, ‘A systematic review of software fault prediction studies’, Expert System Application, Vol. 36, pp. 7346–7354, 2009. doi: 10.1016/j.eswa.2008.10.027.

[3] Hall T, Beecham S, Bowes D, et al, ‘A Systematic Literature Review on Fault Prediction Performance in Software Engineering’ IEEE Transaction of Software Engineering Vol. 38, pp. 1276–1304, 2012. doi: 10.1109/TSE.2011.103

[4] Begel, A., &Nagappan, N., ‘Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study’ in Proceedings of the 1st International symposium on empirical software engineering and measurement IEEE, pp. 255-264, 2007,doi: 10.1109/ESEM.2007.12.

[5] Sharma, P., Sangal, A.L., ‘Building and Testing a Fuzzy Linguistic Assessment Framework for Defect Prediction in ASD Environment Using Process-Based Software Metrics’, Arabian Journal of Science and Engineering, Vol. 45, no.12 pp.10327–10351, 2020 https://doi.org/10.1007/s13369-020-04701-5.

[6] Y. Hassouneh, H. Turabieh, T. Thaher, I. Tumar, H. Chantar and J. Too, ‘Boosted Whale Optimization Algorithm With Natural Selection Operators for Software Fault Prediction’, IEEE Access, Vol. 9, pp. 14239-14258, 2021 doi: 10.1109/ACCESS.2021.3052149.

[7] KapilJuneja, ‘A fuzzy-filtered neuro-fuzzy framework for software fault prediction for inter-version and inter-project evaluation’, Applied Soft Computing Elsevier, Vol. 77, pp. 696-713, 2019.https://doi.org/10.1016/j.asoc.2019.02.008

[8] I. Tumar, Y. Hassouneh, H. Turabieh and T. Thaher, ‘Enhanced Binary Moth Flame Optimization as a Feature Selection Algorithm to Predict Software Fault Prediction’, IEEE Access, Vol. 8, pp. 8041-8055, 2020,doi: 10.1109/ACCESS.2020.2964321.

[9] Wang Geng, ‘Cognitive Deep Neural Network prediction methods for software fault tendency module based on Bound Particles Swarm Optimization’ Cognitive System Research Elsevier, Vol. 52, pp. 12-20, 2018.

[10] C. Jones, ‘Measuring defect potentials and defect removal efficiency’, Journal of Defense Software Engineering, Vol.21, no. 6, pp. 11-13, 2010.

[11] Chidamber S, Kemerer C, ‘A metrics suite for object-oriented design’. IEEE Transaction of Software Engineering, Vol. 20 no. 6 pp.476– 493, 1994.

[12] K. V. J. Padmini, H. M. N. DilumBandara, and I. Perera, ‘Use of software metrics in agile software development process’, Moratuwa Engineering Research Conference (MERCon), pp. 312–317, 2015.

[13] M. Agarwal and P. R. Majumdar, ‘Tracking Scrum projects Tools , Metrics and Myths About Agile’, International Journal of Emerging Technology and Advanced Engineering, Vol. 2, no. 3, pp. 97–104, 2012.

[14] Arisholm, E., Briand, L.C., Johannessen, E.B., ‘A systematic and comprehensive investigation of methods to build and evaluate fault prediction models’, Journal of System and Software, Vol. 83, no. 1, pp. 2–17, 2010.

[15] Briand L., Wüst J., Ikonomovski S., and Lounis H., ‘A Comprehensive Investigation of Quality Factors in Object-Oriented Designs: An Industrial Case Study,’ Technical Report, International Software of Engineering Research Network-98-29, 1998.

[16] T. Sethi, ‘Improved approach for software defect prediction using artificial neural networks’ in the Proceedings of the 5th International Conference on in Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 480-485, 2016.

[17] Ravi Kumar, T., SrinivasaRao, T., ‘Software Defects Prediction based on ANN and Fuzzy logic using Software Metrics’, International Journal of Applied Engineering Research, Vol. 12, no.19, pp. 8509-8517, 2017.