JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Study on Software Fault Detection Prediction in Software Engineering Activities.

Jasmeet Kaur*

Assistant Professor in Computer Science.
Khalsa College(ASR) of Technology and Business Studies,
Mohali –Punjab.

Abstract

In the current era, as our need for complicated software has increased, the demand for excellence of software is becoming more important progressively. Now the software is being utilized almost universally and in every step of the life of human beings and complexity and volume of software systems are increasing with a rapid rate. The software cost such as defect & breakdown may reduce the quality of software and it creates disappointment to the customer. Fault prediction in software plays a vital role in enhancing the software excellence as well as it helps in software testing to decrease the price and time. Software defect prediction typically produce datasets, methods and frameworks which allow software engineers to focus on development activities in terms of defect-prone code, thereby improving software quality and making better use of resources. From 40 years is software defect prediction, where predictions are made to determine where future defects might appear. Since then, there have been many studies and many accomplishments in the area of software defect prediction. But there remain many challenges that face that field of software defect prediction and Prevention approaches. This paper presents a Review on current practices for software fault detection and prevention mechanisms in the software development and brief overview of software defect prediction and its various components, current trends, revisit the challenges of software prediction models and highlight some key of future challenges.

Keywords: literature review, software defect prediction, software defect prediction methods, Prevention approaches, fault datasets

I. Introduction

With the rapid development of computer technology, software applications have expanded to all parts of people's daily lives, creating a situation in which the economy, production, and life are fully dependent on computer software. But software failure can bring about serious or even fatal consequences, especially for high-risk systems. System failure is more often caused by software defects, which are important factors affecting software quality and are potential root causes of errors and failures in the relevant systems [1].

Software quality means to be an error-free product, which will be competent to produce predictable results and able to deliver within the constraints of time and cost. Therefore, a systematic approach for developing high quality software is increased in the competitiveness in today's business world, technology advances, the complexity of the hardware and the changing business requirements. So far, for the fault-prone modules various techniques have been proposed for predicting and forecasting in terms of performance evaluation [2]. Application of Software Fault Detection Prediction models early in the software lifecycle allows practitioners to focus their testing manpower in a manner that the parts identified as "prone to defects" are tested with more

rigor in comparison to other parts of the software system This leads to the reduction of manpower costs during development and also relaxes the maintenance effort [3].

Software quality assurance (SQA) consists of monitoring and controlling the software development process to ensure the desired software quality at a lower cost. It may include the application of formal code inspections, code walkthroughs, software testing, and software fault prediction Software fault prediction aims to facilitate the allocation of limited SQA resources optimally and economically by prior prediction of the fault-proneness of software modules The potential of software fault prediction to identify faulty software modules early in the development life cycle has gained considerable attention over the last two decades. Earlier fault prediction studies used a wide range of classification algorithms to predict the fault-proneness of software modules.

This paper is organized as the overview of the software fault prediction process. present information about the software fault dataset, detail of software metrics, project's fault information, and meta information about the software project, information of methods used for building software fault prediction models.

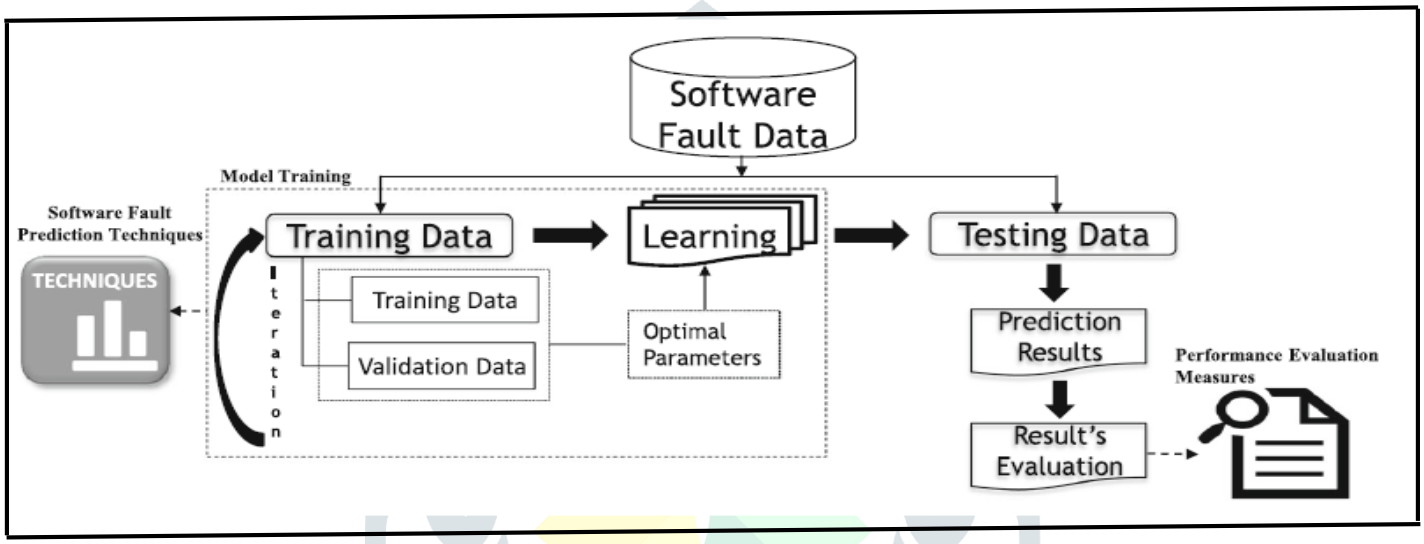

Fig. 1 Software fault prediction process

It's also contained performance evaluation measures. finding of our review study. highlighted some key challenges and future works of software fault prediction and presented the conclusions.

2 Software fault prediction

Software fault prediction aims to predict fault-prone software modules by using some underlying properties of the software project. It is typically performed by training a prediction model using project properties augmented with fault information for a known project, and subsequently using the prediction model to predict faults for unknown projects. Software fault prediction is based on the understanding that if a project developed in an environment leads to faults, then any module developed in the similar environment with similar project characteristics will ends to be faulty The early detection of faulty modules can be useful to streamline the efforts to be applied in the later phases of software development by better focusing quality assurance efforts to those modules.

Figure 1 gives an overview of the software fault prediction process. It can be seen from the figure that three important components of software fault prediction process are: Software fault dataset, software fault prediction techniques, and performance evaluation measures. First, software fault data is collected from software project repositories containing data related to the development cycle of the software project such as source code and change logs, and the fault information is collected from the corresponding fault repositories. Next, values of various software metrics (e.g., LOC, Cyclomatic Complexity etc.) are extracted, which works as independent variables and the required fault information with respect to the fault prediction (e.g., the number of faults,

faulty and non-faulty) work as the dependent variable. Generally, statistical techniques and machine learning techniques are used to build fault prediction models. Finally, the performance of the built fault prediction model is evaluated using different performance evaluation measures such as accuracy, precision, recall, and AUC (Area Under the Curve).

2.1 Software Fault Detection Mechanism

A failure refers to any fault or imperfection in a work activity for a software product or software process cause due to an error, fault or failure. Fault as, a wrong decision while understanding the information given to solve the problems or the application process. A single error can lead to one or more faults and a several faults can lead to failure. To avoid this failure in software products, faults detections activities are carried out in every phase of the software development life cycle based on their need and criticality.

1. Detection Using Automated Static Analysis: - Automated Static Analysis (ASA) detection is mostly performed for the Manual Code analysis, which is one of the oldest practices are still practiced, but automated tools are increasingly used especially for the standard problems related to non-compliance faults possible memory leaks, variable usage etc. Find bugs, Check Style and PMD are some of the commonly used tools in the Java technology and there are many of these tools in all technologies. However, for systems that have compatible source for automatic static analysis detection tools can be used as a hygiene factor and good detection mechanism as any error introduced in the field is highly expensive.

2.Detection Using Graph mining: - Graph Mining is a dynamic control flow based approach that helps identify flaws that may be not crashing in nature. Use graphics calls are reduced by the simplicity in processing. The graph node represents the functions and a function call to another is represented by the edges. Edge weights are entered based on the calling frequencies. The variation in the frequency of call and change in the structure of call are potential failures. If there are problems in the data that is transmitted between the methods could also affect the graph of the named because of its implications

3.Detection Using Classifiers: - Classifiers based on the clustering algorithm and decision tree or neural network can be used to identify abnormal events of normal events for the detections. Classifiers are also formed by labelling defective tracks when a fault is observed. Some classifiers are commonly used Naïve Bayes and bagging. Bayesian classification is a supervised learning method and a statistical method for classification.

Machine learning classifiers have recently introduced in the faults to predict changes in the source files. The classifier is first trained on software development, and then used to predict whether an upcoming change causes an error. Disadvantages of existing classifier-based bug prediction techniques are not enough power for practical use and slow prediction times due to a large number of machines learned functions

4.Detection Using Pattern Mining: - Pattern based detection also the classifier based but uses unique iterative patterns for classification sequential data using the software trace analysis for failure detection. A set of discriminatory features capture

repetitive series of events from the program execution traces first executed. Subsequently, the choice is made to select the best features for classification. Classifier model is trained with these sets of features that will be used to identify the failures.

3.Software Fault Prevention Mechanism

In software development, many faults emerged during the development process. It is a mistake to believe that faults are injected into the beginning of the cycle and removed through the rest of the development process. Fault prevention is a process of quality improvement which aims to identify common causes of faults and change the relevant processes to prevent the type of fault recurrence. It also increases the quality of a software product and reduces overall costs, time and resources.

1.Importance of Fault Prevention: - Faults prevention is an important activity in any software project development cycle. Most software project team focuses on fault detection and correction. Right from the early

stages of the project to prevent faults from being introduced into the product that measure is therefore appropriate to make. Such measures are low cost, the total cost savings achieved due to profit later on stage are quite high compared to the cost of fixing faults. Fault injection methods and processes enable fault prevention knowledge. After practicing this knowledge has improved quality.

2.Activities in Fault Prevention

A. Fault Identification: - Fault can be a pre-planned activities aimed at highlighting the specific faults found. In general, faults can be identified in design review, code inspection, GUI Review, function and unit testing activities performed at different stages of software development life cycle. Once the faults are identified it will be classified using classification approach for the detection.

B. Fault Classification: - Classification of fault can be made using the general Orthogonal Defect Classification (ODC) technique to find the fault group and it type. The ODC technique classifies the faults at the time when fault first occurs and when the fault gets fixed. The first level of ODC classifies the various types of faults in different stages of development requirement like Specification gathering, Logical Design, Testing and Documentation.

C. Fault Analysis: -Fault analysis is the continuous process for the quality improvement using fault data. Fault analysis generally classified in categories blame and direct process improvement efforts in order to attempt to identify possible causes. Root Cause Analysis (RCA) software fault has played a useful role in the analysis. RCA's goal to identify the root cause of faults and flaws that the source is eliminated so is to initiate action. To do this, faults one at a time are analysed. Qualitative analysis is limited only by the limits of human investigative capacities.

D. Fault Prevention: -Fault prevention is an important activity in any software project. Identify the cause of faults and fault prevention objective is to prevent them from recurring. Fault Prevention had suffered in the past to analyse the faults and faults in the future to prevent the occurrence of these types include special operations.

4.Faults Prevention Benefits and Limitations

Fault prevention strategies exist, but reflect a high level of test maturity discipline associated with the testing effort represents the most cost-effective expenditure. To detect errors in the development life cycle from design to implement code specifications require that helps to prevent the escape of errors. Therefore, test strategies can be classified into two different categories as, fault detection technologies and fault prevention technologies. Fault prevention efforts over a period of application development provide major cost and time savings. Thus it is also important, reduces the number of faults for reconstruction brings cost reduction, it is easy to maintain port and reuse makes. The lack of specific domain knowledge, where new and diverse domain software is a need to develop and implement. In many occasions, appropriate quality requirements specified are not in the first place. The inspection operation is labour intensive and requires high skill. Sometimes well-developed quality measurement may not have been identified at design time.

5 Software fault dataset

Software fault dataset that act as training dataset and testing dataset during software fault prediction process mainly consists of three components: set of software metrics, fault information like faults per module, and meta information about project.

The fault information tells about how faults are reported in a software module.In general, three types of fault dataset repositories are available to perform software fault prediction

1) **Private/commercial: -** In this type of repository, neither fault dataset nor source code is available. This type of dataset in maintained and used by the companies within the organizational use.
2) **Partially public/freeware: -** In this type of repository only the project's source code and fault information are available. The metric values are usually not available Therefore, it requires that the user must calculate the metric values from the source code and map them to the available fault information.

3) **Public: -** In this type of repository, the value of metric as well as the fault information both are publicly available (Ex. NASA and PROMISE data repositories). The studies performed using datasets from these repositories can be repeatable.

The fault data are collected during requirements, design, development, and in various testing phases of the software project and are recorded in a database associated with the software's modules Based on the phase of the availability of the fault information, faults can be classified as pre-release faults or post-release faults**.**

TABLE 1.
Information about the programming language and Lines of Code (LOC) and the percentage of defective modules for each dataset is shown in other three columns respectively

Data set	Language	LOC	# sample (positive,negative)
CM1	C	20k	505(0.095,0.905)
KC1	C++	43k	2107(0.154,0.845)
KC2	Java	18k	522(0.201,0.798)
KC3	Java	18k	458(0.093,0.906)
KC4	Perl	25k	125(0.6,0.4)
MC1	C++	63k	9466(0.007,0.992)
MC2	C	6k	161(0.322,0.677)
PC1	C	40k	1107(0.068,0.931)
PC2	C	26k	5589(0.004,0.995)
PC3	C	40k	1563(0.102,0.898)
PC4	C	36k	1458(0.122,0.878)
PC5	C++	16k	17186(0.030,0.970)
JM1	C	315k	10878(0.19,0.81)
MW1	C	8k	403(0.08,0.92)

TABLE 1. The detailed information of NASA datasets

5.1. Software metrics: - For an effective and efficient software quality assurance process, developers often need to estimate the quality of the software artefacts currently under development. For this purpose, software metrics have been introduced. By using metrics, a software project can be quantitatively analysed and its quality can be evaluated. Generally, each software metric is related to some functional properties of the software project such as coupling, cohesion, inheritance, code change, etc., and is used to indicate an external quality attribute such as reliability, testability, or fault-proneness

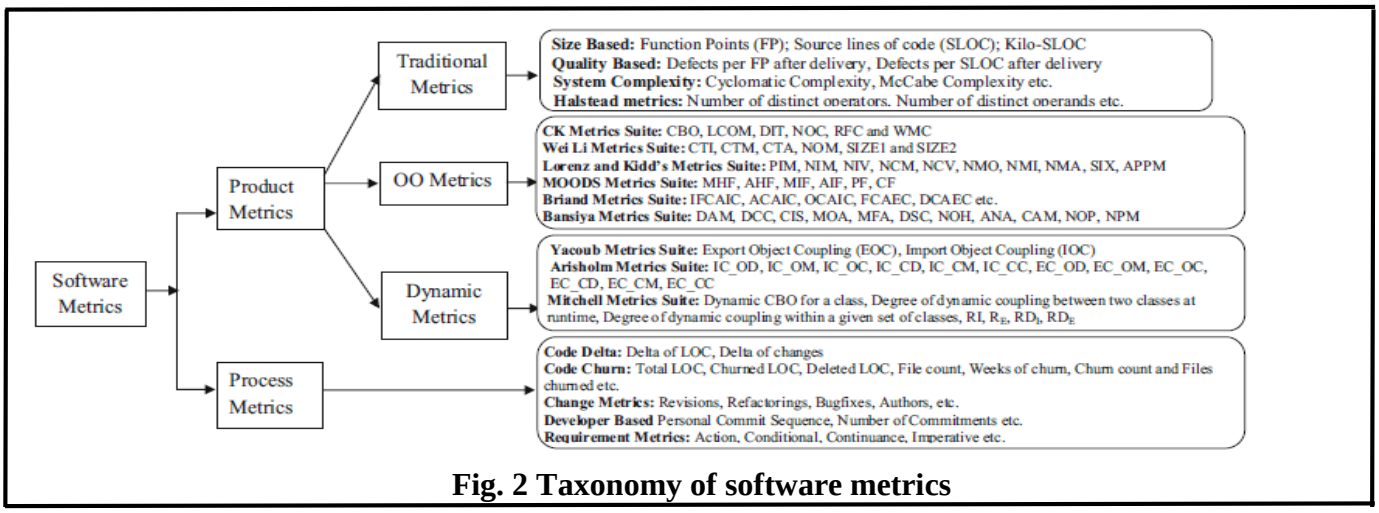

Fig. 2 Taxonomy of software metrics

5.2 Meta information about project: - Meta information about project contained the information of various characteristics of software project. It consists various set of information's such as the domain of software development, the number of revisions software had, etc. as well as consist information of the quality of the fault dataset used to build fault prediction model. Figure shows various attributes of the meta information about the project.

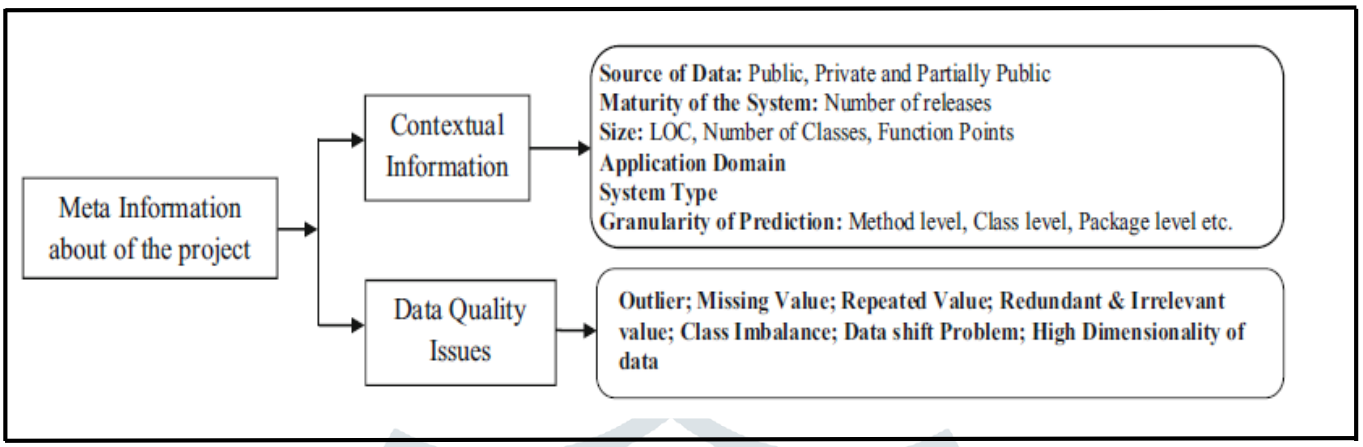

Fig. 3 Meta information of the software project

6. Methods to build software fault prediction model

Various techniques for software fault prediction are available in the literature. We have performed an extensive study of the available software fault prediction techniques and based on the analysis of these techniques a taxonomic classification has been proposed, as shown in Fig Figure shows various schemes that can be used for software fault prediction. A software fault prediction model can be built using the training and testing datasets from the same release of the software project (Intra-release fault prediction), from different releases of the software project (Inter-release fault prediction) or from the different software projects (cross project fault prediction). Similarly, a software fault prediction model can be used to classify

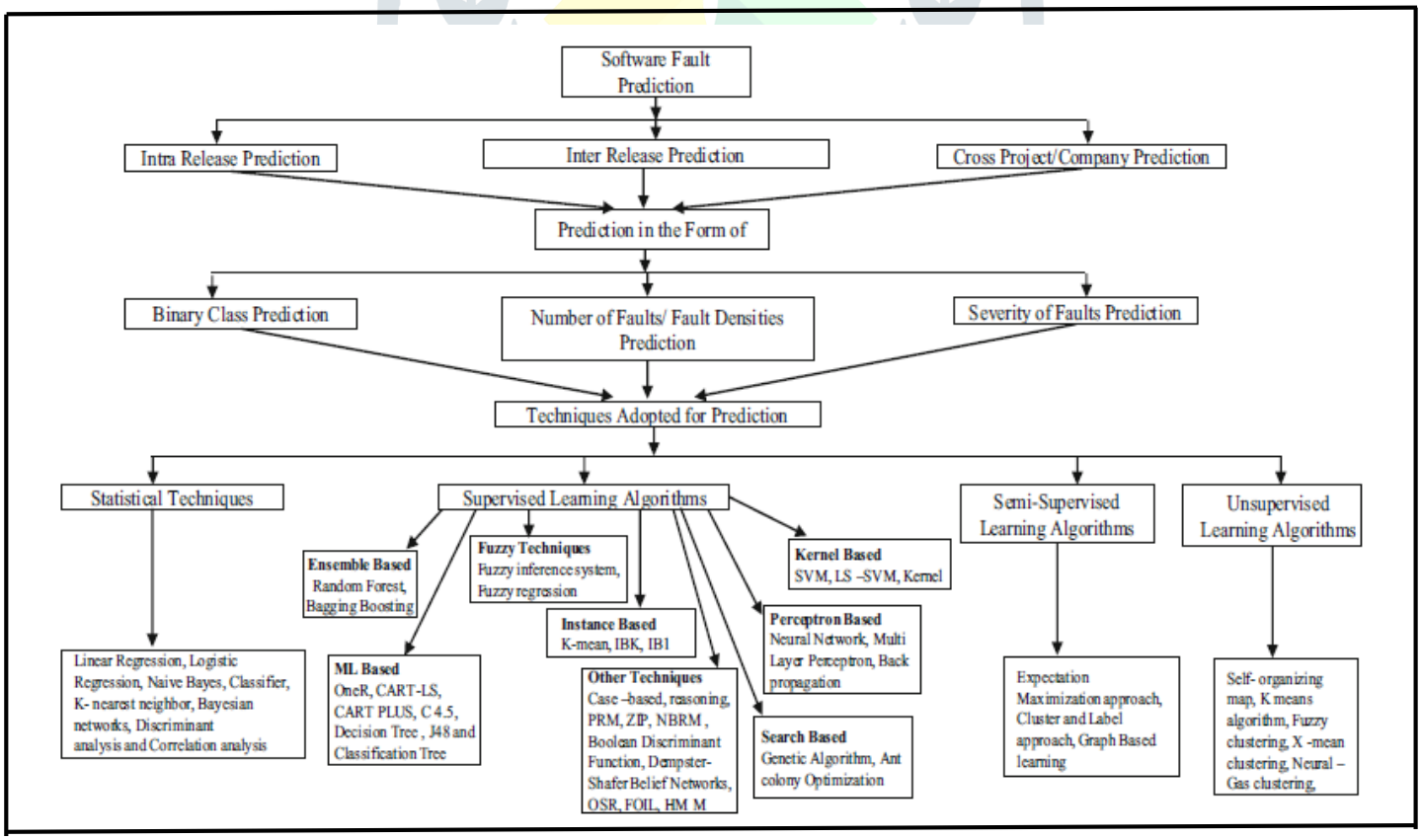

Fig. 4 Taxonomy of software fault prediction techniques

software modules into faulty or non-faulty categories (binary class classification), to predict the number of faults in a software module, or to predict the severity of the faults. Various machine learning and statistical techniques can be used to build software fault prediction models. The different categories of software fault prediction techniques.

6. Performance evaluation measures

Various performance evaluation measures have been reported in the literature to evaluate the effectiveness of fault prediction models. In a broad way, evaluation measures can be classified into two categories: Numeric measures and Graphical measures. Figure illustrates taxonomy of performance evaluation measures. Numeric performance evaluation measures mainly include accuracy, specificity, f-measure, G-means, false negative rate, false positive rate, precision, recall, j-coefficient, mean absolute error, and root mean square error. Graphical performance evaluation measures mainly include ROC curve, precision-recall curve, and cost curve

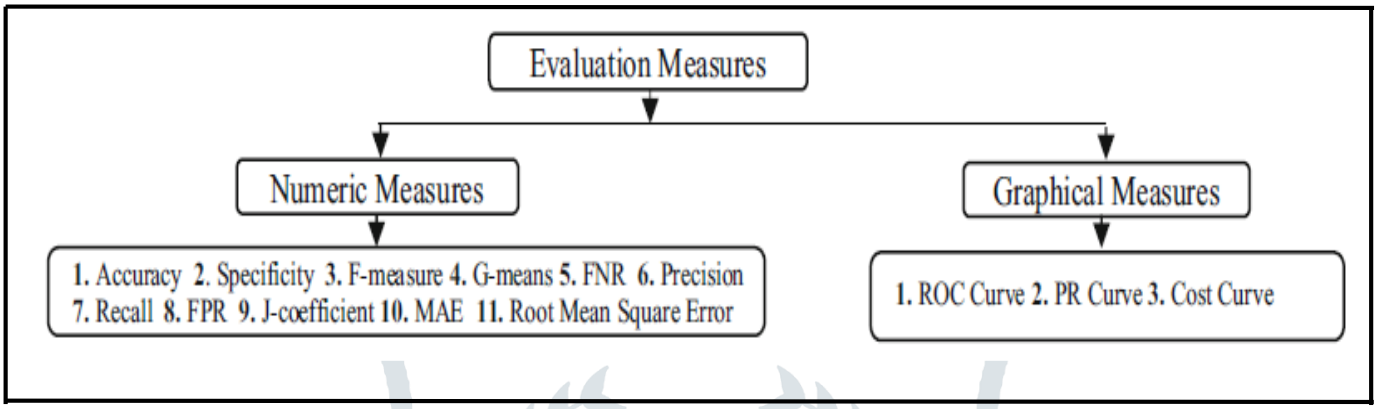

Fig. 5 Taxonomy of performance evaluation measures

7. Related Work

Software fault prediction has been an important research topic in the software engineering field for more than three decades, increasingly catching the interest of researchers. In the literature, authors have addressed the software fault prediction (SFP) problem. Over the preceding two decades, software researchers have shown great prominence in fault prediction studies, as evident from work dealing with the development of fault prediction models. Examining the Predictive Capability of Advanced Software Fault Prediction Models were presented by Pooja Sharma, Amrit Lal Sangal [4] Li Zhiqiang et al. [5], and Radjenovic et al. [6], in the context of advanced models for software fault prediction, in which authors have used information related to product and process metrics. The models for investigation were built based on five different scenarios as discussed in Section 4. Scenario-1: simple model (consists of product metrics only); scenario-2: advanced model-1 (product metrics + NR process metric); scenario-3: advanced model-2 (product metrics + NDC process metric); scenario-4: advanced model-3 (product metrics + NML process metric); scenario-4: advanced model-4 (product metrics + NDPV process metric). The various base classifiers used to predict the performance of proposed models are DT, MLP, RT, SVM and NB. The study has been conducted on thirty-two open-source code projects extracted from the Promise, Jira and Bug repositories. The results show that among base classifiers the MLP based base classifier captures high average accuracy (87%), average ROC(AUC) (79%), average *F*-score (83%) and least RMSE error (0.12) for advanced model-2 constructed using (product + NDC metrics) from Promise repository as compared to other advanced models, i.e., advanced model-1, advanced model-3 and advanced model-4, respectively.

Defect Prediction: Accomplishments and Future Challenges by Yasutaka Kamei, Emad Shihab [7] . in the context of highlighted, there have been many papers that addressed challenges related to data, metrics, model building and performance evaluation. At the same time, particular initiatives and works have had a profound impact on the area of software defect prediction, which we listed as game changers. OSS projects providing

rich, extensive, and readily available software repositories; the PROMISE repository providing SE researchers with a common platform to share datasets; the SZZ algorithm automatically extracting whether or not a change introduces a defect from VCSs, which in turn dramatically accelerated the speed of research, especially for JIT defect prediction; tools such as Weka and R providing a wide variety of data pre processing

Defect Prediction Technology of software Based on Deep Neural Network[8] by analysing the characteristics of aerospace software, this paper proposes a software defect prediction method based on process measurement. ,is study proposed to combine the software process with NASA software data metrics to form a metrics set to design a measurement set oriented to the characteristics of aerospace software. we established a deep neural network through an auto encoder network model to form software defect prediction model. But this method is now only suitable for aerospace engineering software and requires manual testing of the two versions of the data as a training set to achieve better results. rough experimental analysis, the average accuracy of the prediction technology reaches 90%, which proves that the aerospace software defect prediction technology proposed in.

Defect Prediction Using ANN [9] researchers explored that the ANN is considered as one of the widely used supervised machine learning techniques to predict the defects at early stages of SDLC. It has been concluded that ANN has wide scope for software defect prediction especially when used with a hybrid approach. *e performance of proposed techniques was evaluated in terms of various measures, calculated from confusion matrix and compared with conventional base classifiers as well as with proposed optimized techniques.

Fault-Detection and Fault-Correction Processes in Modeling Software by Qiuying Li and Hoang Pham. [10] In this paper, the problem of modeling software fault-detection processes and fault correction processes together with imperfect debugging and testing coverage has been investigated. From the viewpoint of fault amount instead of fault correction time delay, a new software reliability growth models. model is applied to two kinds of data sets: one that contains information not only of detected fault numbers but also corrected fault numbers, and another that contains only detected fault numbers. It should be noted that, though adding more parameters makes the software reliability model more complicated and the task of parameter estimation more difficult, by automating the calculations using software tools it is not a critical problem.

Jinyin Chen , Keke Hu, Yitao Yang , Yi Liu , Qi Xuan [11] propose a collective training mechanism for defect prediction . There are two main contributions, including source data expan- sion an d adaptive weighting, in this work. First, we use several normalization methods to capture different information of source projects. Second, we comprehensively integrate the information into a collective classifier by an adaptive weighting process to get better prediction results.

Devika S, Lekshmy P L [12] It helps to recognize modules that are bug- free and bug-prone in a software module. Machine learning techniques for both classification and determination are used for the purpose of software fault prediction. Software Fault Prediction is carried out prior to testing process without executing the source code, instead vital characteristics of software is taken into consideration. This early identification of faults can help software engineers to reduce the risk of system failure. the faulty modules identified by both testing phase and software prediction models where the same. Some of the issues such as imbalance of software modules, high data dimensionalities, noisy data, etc. must be dealt carefully. The functioning of Software Fault Prediction process highly depends on the use of base learners and uniqueness of software fault datasets. The above two challenges raise the need of using multiple learning techniques for predicting faults in a given software system.

8.Conclusions

The paper reviewed works related to various activities of software fault prediction such as software metrics, fault prediction techniques, data quality issues, and performance evaluation measures. From this extensive review, it can be concluded that more studies proposing and validating new software metrics sets by exploring the developer's properties, cache history and location of faults, and other software development process related properties are needed. The future studies can try to build fault prediction models for cross-project prediction that can be useful for the organizations with insufficient fault project histories. The results of reviews performed in this work revealed that the performance of the different fault prediction

techniques vary with different datasets.

References

1] X. Zhang, X. Chen, J. Wang, Z. Zhan, and J. Li, "Verifiable privacy-preserving single-layer perceptron training scheme in cloud computing," Soft Computing, vol. 22, no. 23, pp. 7719–7732, 2018.

[2] B. Dhanalaxmi, Dr.G. Apparao Naidu, Dr.K. Anuradha, "A Review on Software Fault Detection and Prevention Mechanism in Software Development Activities," IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 17, Issue 6, PP. 25-30, Ver. V Nov – Dec. 2015.

[3] Le Hoang Son, Nakul Pritam , Manju Khari , Raghvendra Kumar ,Pham Thi Minh Phuong and Pham Huy Thong, "Empirical Study of Software Defect Prediction: A Systematic Mapping," MDPI journals ,2019.

[4] Pooja Sharma, Amrit Lal Sangal, "Examining the Predictive Capability of Advanced Software Fault Prediction Models – An Experimental Investigation Using Combination Metrics," e-Informatica Software Engineering Journal, Volume 16, Issue 1, pages: 220104, DOI 10.37190/e-Inf220104, 2022.

[5] Z. Li, X.Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," *Iet Software*, Vol. 12, No. 3, pp. 161–175. 2018.

[6] F. Matloob, T.M. Ghazal, N. Taleb, S. Aftab, M. Ahmad et al., "Software defect prediction using ensemble learning: A systematic literature review," *IEEE Access*, 2021.

[7] Yasutaka Kamei, Emad Shihab., "Defect Prediction: Accomplishments and Future Challenges," *Yasutaka Kamei on 31 December 2017.*

[8] Tianwen Yao , Ben Zhang, Jun Peng, Zhiqiang Han, Zhaobing Yang, Zhi Zhang, and Bo Zhang," Defect Prediction Technology of Aerospace Software Based on Deep Neural Network and Process Measurement" Hindawi Mathematical Problems in Engineering Volume 2022, Article ID 1276830, 8 pages https://doi.org/10.1155/2022/1276830 , Published 1 February 2022.

[9] Muhammad Adnan Khan, Nouh Sabri Elmitwally, Sagheer Abbas, Shabib Aftab, Munir Ahmad, Muhammad Fayaz , and Faheem Khan" Software Defect Prediction Using Artificial Neural Networks: A Systematic Literature Review" Hindawi Scientific Programming, Article ID 2117339, 10 pages https://doi.org/10.1155/2022/2117339, Published 12 May 2022.

[10] Qiuying Li 1,2,* and Hoang Pham," Modeling Software Fault-Detection and Fault-Correction Processes by Considering the Dependencies between Fault Amounts", https://www.mdpi.com/journal/applsci, Appl.Sci.,11,6998.https://doi.org/10.3390/app11156998, 2021.

[11] Jinyin Chen , Keke Hu, Yitao Yang , Yi Liu , Qi Xuan " Collective transfer learning for defect prediction" www.elsevier.com/locate/neucom, 2019 .

[12] Devika S, Lekshmy P L," Best Suited Machine Learning Techniques for Software Fault Prediction" International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020.