CHAPTER 2

LITERATURE REVIEW

2.1 OVERVIEW

Software Engineering is the profession to examine, plan, develop and maintain the software. Conventional software development methods concentrate mainly on software correctness, which involves performance problem in the stage of development process. Quality explains the success of the software, it tells about, how well the software is planned and it says about the fulfilment of the requirement design, also it is describes the cost effective, mean time failures and security predictions. The terms like Errors, faults, failures and defects were also connected. Error is made by human mistakes, which creates wrong output. Error causes the software failure, so the software doesn’t meet the requirements. This failure causes defect. Software quality’s primary aspect is to foresee the fault in object oriented software systems. It is accomplished by various models that were suggested in software verification and validation.

2.2 FAULT PREDICTION

The fault prediction is an effective methodology to enhance the product’s quality so, in order to enhance the quality, we need to detect the error or fault as quick as possible. To define and to improve the development
of product, we have various prediction approaches, such as correction cost prediction, test effort prediction, software fault prediction (SFP), security prediction and so on. This chapter examines the various methods and metrics, which helps to enhance the quality of the object oriented software.

2.2.1 Statistics Based Approach

Zero-inflated poisson regression model and Poisson regression method is utilized to foresee the fault in the system and make the system more reliable. Our work examines the full-scale industries software systems. In order to depreciate the zeros of dataset variable values, Poisson regression model is utilized. The file metrics adds the length and number of source codes, comments length in source code. The output of distinguishing the two models confirms that the zero-inflated Poisson regression gives better performance.

Fault prediction in an industrial telecommunication system is proposed in (Denaro et al. 2003), they make use of logistic regression in telecommunication system through class level metrics. We can’t compute the CBO and LCOM metrics, just through learning with the help of automated tool, so, the metrics predicts coupling among the classes and in these methods, we have deficiency of cohesion. The multivariate model doesn’t offer much merits when distinguished with lines of code metrics and also they explain fault proneness. Our work is executed initially in C and included with an object oriented systems. This development and bad OO design creates correlated issues. Grouping of a header and CPP file is a module.

The software fault-proneness of modules in antenna configuration software is computed in (Denaro, G. 2000), which utilizes logistic regression and 37 metrics. R2 was the evaluation parameter in our work. The
author explains that we have a correlation among the static metrics and fault-proneness.

Software quality is predicted in (Khoshgoftaar et al. 2000) with the help of case-based reasoning and eight method level metrics for a command-control-communication system established with ADA language. Method level metrics are lines of code, Halstead, and McCabe metrics. Type-I and Type-II error were utilized as performance analysis metrics and here prediction model was helpful.

Principal component analysis for feature selection is utilized initially in (Xu et al. 2000) and then they enforce fuzzy nonlinear regression (FNR) in order to foresee the faults on a huge telecommunications system, which is established with Protel language. 24 method level metrics were utilized and four execution metrics can be gathered with EMERALD tool. Performance analysis shows average absolute error. Further, they stated that the fuzzy nonlinear regression method is a motivating technology for foreseeing the faults early. The fuzzy values were generated by FNR modelling and it foresees an interval for every module. This interval gives fault’s count for the next release. In our work, a module was determined as a set of source code files.

Finite mixture model analysis with Expectation–Maximization (EM) algorithm is enforced in (Guo and Lyu. 2000), in order to estimate the software quality on a medical imaging system which is established with Pascal and Fortran languages. Method level metrics like Halstead, McCabe, Belady’s bandwidth, Jensen’s program length were utilized and Type-I, Type-II, and overall misclassification rates were utilized as a performance evaluation metrics. Our study states that this approach can foresee the quality even if fault
data aren’t finished. Also, the best value for Type-II error was 13% and which is same as the value generated with discriminate analysis.

Zero-inflated poisson regression model (ZIP) and file level metrics on two large scale software applications, helps to foresee the faults in (Khoshgoftaar et al. 2001). File metrics includes number of source code inspections before system tests, the length of source code prior to system tests, the length of comments in source code prior to coding (auto-generated code), and the length of comments in source code prior to system tests. Whereas performance metrics includes average absolute error (AAE) and Average Relative Error (ARE). In our work, poisson regression model (PRM) and ZIP models were examined. When distinguished with PRM, ZIP gives good output. Furthermore, AAE and ARE values of ZIP was smaller when compared to PRM’s error results. The zero inflated Poisson (ZIP) model is one way to allow for over dispersion. This model assumes that the sample is a “mixture” of two sorts of individuals: one group whose counts are generated by the standard Poisson regression model, and another group (call them the absolute zero group) who have zero probability of a count greater than 0. Observed values of 0 could come from either group.

Boolean Discriminate Functions (BDF) and logistic regression functions (LRF) for software quality prediction were examined in (Schneidewind 2001) with the help of six method level metrics on a spacecraft software dataset. Type-I error, Type-II error, overall misclassification rate, and the rate of correctly classified non-faulty modules(LQC) parameters helps in performance evaluation metrics. When compared to LRF’s performance, BDF’s performance was good and the inspection cost raises when the LRF and BDF methods were linked. Logistic regression can be binomial, ordinal or multinomial. Binomial or binary logistic regression deals with situations in
which the observed outcome for a dependent variable can have only two possible types, "0" and "1"

The fault-prone classes are foreseen in (Emam et al. 2001), which assist the logistic regression and class level metrics on a commercial Java application. Two metrics from Chidamber–Kemerer metrics suite and eight metrics from Briand’s metrics suite were chosen. Furthermore, size metric were utilized along with these metrics. In our study, J coefficient was the performance evaluation metric. They state that the inheritance depth and export coupling assist the metrics to recognize the fault-prone classes. In our earlier work, they reported that inheritance depth isn’t desirable. The design quality of this software is low and it will affect the output of our work. The export coupling metric is helpful for our work, when compared with inheritance depth metric.

Generalized Boolean Discriminant Functions (GBDF) technique for software quality classification on a military command-control-communication system is enforced in (Khoshgoftaar, T. M. 2002), with the help of eight method level metrics (Halstead and McCabe). Type-I error and Type-II error parameters were assisted as performance evaluation metrics. Khoshgoftaar stated that GBDF gives good output, when distinguished with BDF. Type-I error of BDF was very high and inspection cost of BDF model was high. Discriminant analysis works by creating one or more linear combinations of predictors, creating a new latent variable for each function. These functions are called discriminant functions. The number of functions possible is either \( N_g - 1 \) where \( N_g = \) number of groups, or \( p \) (the number of predictors), whichever is smaller. The first function created maximizes the differences between groups on that function. The second function maximizes differences on that function, but also must not be correlated with the previous
function. This continues with subsequent functions with the requirement that the new function not be correlated with any of the previous functions.

### 2.2.2 Machine Learning Approach

The java based object oriented software system fault prediction performance according to the Density Based spatial Clustering of Application with Noise (DBSCAN) is explained in (Supreet Kaur, and Dinesh Kumar 2012). This DBSCAN computes java based object oriented systems with accurate value. The metric based approach is utilized in our work for prediction. Also the performance of fault proneness of classes is computed for C++ language based components which works on NASA metrics data program data repository. The metrics based approach is better in minimizing the attribute sets, which is accomplished from the output exactly. With this attribute set, the accuracy of the prediction raises from 85.3712% to 90.6114%. DBSCAN result stated that the accuracy of prediction of 58.63% for the minimized set attributes. The density based clustering gives probability detection for non-faulty model equal to 1 and false alarm probability for faulty model is equal to 0, in both cases. So it results that DBSCAN are acceptable for prediction based on fault proneness.

Examining different machine learning techniques for software defect predication is done in (Pooja Paramshetti and Phalke 2012). Here, it is acquired that the software defect is certainly a critical problem in software engineering. Software Defect Module Prediction with the help of various machine learning techniques enhances the quality of software development process. The software manager efficiently allots the resources, with the help of this technique. The author analyzed the merits and demerits of artificial neural network, Support vector machine, Decision tree, association rule and Clustering machine learning techniques, for foreseeing the defects.
A learning-to-rank approach is brought-in in (Yang et al. 2015), to build the software defect prediction models by exactly enhancing the performance ranking. This model is built on the previous work, additionally; the concept of exactly enhancing the model performance computes the merits of the benefit software defect prediction model construction. The work involves two aspects: one is a new application of the learning-to-rank approach to real-world data sets for software defect prediction, and the other is a complete assessment and comparison of the learning-to-rank method against other algorithms that have been utilized for foreseeing the order of software modules based on the foreseen defects. Our experimental study explains the efficiency of directly optimizing the model performance measure for the learning-to-rank approach to build the construct defect prediction models for the ranking task.

NASA dataset was utilized in (Baljinder et al. 2014), where the author derived same results to the prior study, i.e., the consequences that the classification techniques have appeared to be minimal. Further, the author duplicated the procedure to two new datasets: (a) the cleaned version of the NASA dataset and (b) the PROMISE dataset has open source software established in different kinds of settings e.g., Apache, GNU). The output in these new datasets perfectly shows the statistically distinct separation of groups of techniques, i.e., the option of classification technique has consequence on the performance of defect prediction models. Certainly, dissimilar to the previous research, our output explains that few classification techniques be apt to generate the defect prediction models which beats the rest of the approaches.

The faults with genetic algorithm on KC2 dataset is foreseen (Kaminsky and Boetticher 2004) with the help of 21 method level metrics. They explained the effect of data equalization in our study and T-test were utilized for
performance evaluation. They classified the data starved domains into two groups: Explicit and Implicit.

General Regression Neural Networks (GRNN) technique is enforced in (Kanmani et al. 2004) with the help of 64 class level metrics on student projects which is established in Pondicherry Engineering College for software quality prediction. Principal component analysis were utilized for feature selection and computation parameters were correlation coefficient (r), R2, averagesquare error, average absolute error, maximum absolute error, and minimum absolute error parameters. They stated that the GRNN technique gives good output for fault prediction.

Software quality is foreseen in (Xing et al. 2005) with the help of Support Vector Machines (SVM) and 11 method level metrics (Halstead, McCabe, Jensen’s program length, and Belady’s bandwidth) on a medical imaging software. Type-I error and Type-II error were utilized to compute the performance of model. They stated that SVM’s performance is good when compared with the than quadratic discriminate analysis and classification tree. Nevertheless, papers written in 2007 states that SVM doesn’t work well on the public datasets and next section will describe them clearly.

J48 and KStar algorithms examines the effect of module size on fault prediction in (Koru and Liu 2005). F-measure was utilized for computing the performance and method level and class level metrics were examined on public NASA datasets. They recommended with the help of fault predictors on huge components and they considered utilizing class level metrics rather than the level metrics on small components. A component is detected as large or small win terms of lines of code. Static measures in small components are near to 0, so it doesn’t suit in such datasets. The best performance was accomplished with J48 algorithm and Bayesian Networks, artificial neural
networks, and Support Vector Machines doesn’t work well on public datasets. Performances of models weren’t well enough when we have very small components in datasets. They reported that the machine learning algorithms in WEKA tool are neither encouraging and nor frustrating.

C4.5 decision tree, discriminate analysis, case based reasoning, and logistic regression on two embedded software which configures wireless products determines the new three group software quality classification technique in (Khoshgoftaar et al. 2005). Five file level metrics were enforced and the performance evaluation metrics were anticipated cost of misclassification. This states that the three groups like high, medium, and low labels gives a motivating performance for fault prediction and three-group classification technique allows classifiers to divide the modules into three classes without modifying their designs.

K-Star, and Random Forests on public NASA datasets constructs the fault prediction models in (Koru and Liu 2005b), and they utilized methods and class level metrics for construction. AS a performance evaluation metric, F-measure was chosen. The method level metrics is there in KC1 dataset and they transmit them into class level ones with the help of minimum, maximum, average and sum operations. Hence, 21 method level metrics were transformed into 84 (21 * 4 = 84) class level metrics. This states that huge modules had higher F-measure values for J48, K-Star and Random Forests algorithms. F-measure was 0.65 when they enforce the class level metrics and while they were selected as a method level metrics, F-measure was 0.40. Hence, this reports that the class level metrics enhances the model performance, but identification of faults was at class level rather than the model level.
2.2.3 Statistical and Machine Learning Approach

Techniques like CART-LS (Classification and regression trees tools – least square), CART-LAD (Classification and regression trees tools – Least absolute deviation), S-PLUS (S language –based statistical modeling tool), Multiple linear regression, Artificial neural networks, case based reasoning, were distinguished in (Taghi et al. 2003). In our work, they gathered software metrics from four releases of huge telecommunications systems. In order to detect the accuracy of various prediction models we make use of metrics like performance, average absolute value and average relative errors. Model was generated with the help of RAW original software metrics and their principle components.

The related technologies about classifiers and distribution model were examined in (Wanjiang et al. 2014). For GUI projects, various software defects were gathered, and the paper utilizes lots of classifier algorithms to get defect classification table, then it further enforces the mathematical methods to provide the distribution of this kind of software project defects. If the distribution of the software defects were accepted with respect to the defects classification is recognized as fault injection method, in order to reproduce the software fault, and examine the accelerated test method under specific defects distribution, which efficiently enhances the software test coverage, reduce test time, and reduce cost of test.

Software systems proceed to play a vital role in our daily lives, by creating the quality of software systems as a significant problem. Hence, important research work concentrates much on prioritization the software quality assurance efforts. Software Defect Prediction (SDP) is nothing but a One line of work which has been acquiring an enormous amount of interest, where the predictions were done to define whether the faults repeat in future.
This report shows that, in the past decade, more than 100 papers were published on SDP. However, the practical adoption of SDP to date is limited (Emad Shihab 2014).

SPRINT and CART methods with 28 method level metrics (24 product metrics and four execution metrics) on a huge telecommunications system for software quality classification is enforced in (Khoshgoftaar and Seliya 2002). SPRINT is a classification tree algorithm and CART is a decision tree algorithm. Performance evaluation metrics were Type-I error, Type-II error and overall misclassification rate. This report shows that the SPRINT algorithm had lower Type-I error and the model based on SPRINT was robust. Nevertheless, the tree structure of SPRINT is difficult when compared with the CART tree.

The multi-layer perception and method/class level metrics on a research prototype foresees the quality of the software in (Pizzi et al. 2002). As a performance evaluation metric, Accuracy parameter was utilized. This report shows that before multi-layer perception is enforced, the median-adjusted class labels, is considered as an effective pre-processing technique.

Tree based software quality prediction models on a huge telecommunications system is utilized in (Khoshgoftaar and Seliya 2002b) and they enforce the design metrics. Along with this, CART-LS (least squares), S-PLUS, and CART-LAD (least absolute deviations) were also examined in our work. This report states that the CART-LAD’s accuracy is good and its interpretation is easy. Nevertheless, SPLUS’s doesn’t fit this study and its classification tree is difficult to understand. So, CART-LAD is suggested for software quality prediction.
Software quality based on General Regression Neural Network (GRNN) and Ward Neural Networks is predicted in (Denaro et al. 2003). They make use of level metrics and shows that the GRNN gives good output when compared with Ward networks. Evaluation parameters which are utilized here are: $R^2$, $r$, average square error, average absolute error, minimum absolute error, and maximum absolute error.

CART-LS, CART-LAD, S-PLUS, multiple linear regression, neural networks, and case based reasoning on a huge telecommunications system is enforced in (Khoshgoftaar and Seliya 2003). For this survey, 24 product and four execution metrics were independent variables. The two-way ANOVA randomized complete block design model is utilized as an experimental design approach and multiple-pair wise comparison for performance ranking. Best performance was accomplished with CART-LAD algorithm and the worst one was S-PLUS. Additionally, they report that the principal component analysis doesn’t enhance the models performance, but it eliminates the correlation among the metrics and the model becomes more robust.

Dempster–Shafer Belief Networks and method level metrics predicts the fault-prone modules on NASA’s KC2 project in (Guo et al. 2003). Probability of detection, effort, and accuracy were considered as Performance evaluation metrics. From the report it is stated that the prediction accuracy of this network is greater when compared with logistic regression and discriminate analysis. And this network is cost-effective when compared with ROCKY from the effort perspective. With logistic regression procedure the best metrics set is selected. The optimal prediction on KC2 dataset was accomplished if the metrics count is between two and four.
In order to predict the fault-prone modules, logistic regression with method level metrics on antenna configuration system, Apache 1.3, and Apache 2.0 software is utilized in (Denaro et al. 2003). R2, completeness, completeness of faulty modules, and correctness of faulty modules were considered as Performance evaluation metrics. The report states that the logistic regression with cross-validation is an efficient approach for software fault prediction on industrial projects. Additionally, they recommend to use the cross-validation technique if we have a restricted data.

Naïve Bayes algorithm on public datasets locating in PROMISE repository is enforced in (Menzies et al. 2004). As a performance evaluation metrics, probability of detection (PD) and probability of false alarm (PF) were selected and here Method level metrics were also utilized. The report states that the accuracy isn’t considered as an appropriate parameter for computing and Naive Bayes gives good performance when compared with J48 algorithm. Additionally, they state that PD on KC1 dataset was 55% and PD for Fagan inspections was in the range of 3% and 65%. For industrial inspections, PD for Fagan inspections should be in the range of 13% and 30%. Hence, they recommend software fault prediction activity along with the inspection quality assurance activity.

The software quality classification techniques like logistic regression, case based reasoning, classification and regression trees (CART), tree based classification with S-PLUS, Spring-Sliq, C4.5 and Tree disc were used in (Khoshgoftaar and Seliya 2004), with the help of product metrics and four execution metrics on a huge telecommunications system. As a performance evaluation parameter, the expected cost of misclassification metric was considered. In terms of various versions of software system, the performance of the models gets modified. This report states that the data and
system characteristics affect the performance of prediction models in software engineering.

The seven product metrics computed with MATRIX analyzer is utilized for artificial neural networks for quality prediction on a huge telecommunications system established with C language in (Wang et al. 2004). Their target was to enhance the understandability of neural networks according to the quality prediction models and so they enforce Clustering Genetic Algorithm (CGA). This algorithm mines the rules from artificial neural networks and accuracy parameter is utilized for performance analysis. Though the accuracy of the prediction is done with the rule set of CGA, which is lesser when compared with the accuracy of the neural networks based prediction. The output shows that it is more understandable, if the rule set of CGA was utilized.

The linear regression, pace regression, support vector regression, neural network for continuous goal field, support vector logistic regression, neural network for discrete goal field, Naive Bayes, instance based learning (IBL), J48, and 1-R techniques is used in (Challagulla et al. 2005), with the help of method level metrics on public NASA datasets for software fault prediction. The average absolute error is considered as Performance evaluation metric. The report states that, we have no method to give the best performance in this entire datasets. With respect to accuracy parameter, IBR and 1-R performance was good when compared with the other algorithms and the report states that the principal component analysis doesn’t give any merits. Additionally, the size and complexity metric were examined, but they aren’t sufficient for fault prediction new metrics as required.

The logistic regression, linear regression, decision trees, and neural networks on Mozillaopen source project were utilized in order to certify
the object-oriented metrics for fault prediction (Gyimothy et al. 2005). The completeness, correctness, and precision were considered as performance evaluation metrics, along with this, Class level metrics were utilized. The report states that coupling among object classes (CBO) metric assist in fault prediction, four methods give common results, when compared with lines of code metric, multivariate models were helpful, depth of inheritance tree (DIT) metric doesn’t suit fault prediction, and number of children (NOC) metric shouldn’t be utilized.

The negative binomial regression model on two industrial systems were foreseen for the location and number of faults and few metrics utilizes programming language, age of file, and file change status. Performance evaluation metric was accuracy. The report states that the accuracy of general performance was 84% and the simplified model’s accuracy was 73%.

The accuracy of early fault prediction in modified code is examined in (Tomaszewski et al. 2005). The regression techniques and method/class level metrics were enforced on a huge telecommunications system and performance evaluation parameter R2 (determination coefficient). The report states that the models constructed after the execution of the system were 34% more accurate, when compared with the models which are constructed before the system is executed. Modified classes metric's size can be computed after the system implementation and hence, the performance of models enhances by utilizing this metrics. Performance values before and after the system implementation is nearly same, if this metrics isn’t chosen. When distinguished with univariate regression models, the Multivariate models were 5% better, but this can be neglected. The univariate regression models are recommended for fault prediction, since they are more stable and they don’t have multico linearity feature. When we have correlation among the variables,
this feature occurs and it can be eliminated by some techniques like principal component analysis.

2.2.4 Statistical models vs. Expert estimation

The measure for object oriented software systems by raising the cohesion of classes and less coupling among the methods is suggested in (Marcus et al. 2008). Unstructured information is implanted in the source code to use in this process. Here the Comments and identifiers were considered as the unstructured information. For fault prediction in object oriented software systems, the measure C3 (conceptual cohesion of classes) is utilized. To determine the cohesion of the class, different paths were there such as structural metrics, semantic metrics and so on. By using the latent semantic indexing, the identifiers in object oriented systems Information retrieval approaches were utilized. The fault prediction is explained in details in our work as a case study. The principle component analysis of the metric data is discussed in our 1st case study. The fault prediction in classes is given in the next case study. At last, the output shows that the conceptual cohesion of classes C3 is a precious complement in a number of combinations with other structural cohesion metrics according to the regression analyses results. It assists the prediction of software system.

The software fault detection generally works upon the machine learning approaches, which is noticed in (Rohit et al. 2014) and with the help of the NASA’s public datasets, we can foresee the software faults. Public Datasets were generally pin-pointed in PROMISE and NASA MDP (Metrics Data Program) repositories and they can spread easily. Method Level metrics and Class Level metrics were significant utilized. Machine learning models have good performances and feature, when compared with the Statistical methods or expert opinion. So, the machine learning models were utilized
commonly and these models raises the consumption of public datasets for fault prediction in future.

The class level metrics on an industrial telecommunications system for fault prediction is enforced for logistic regression in (Denaro et al. 2003). In our work, CK metrics except coupling among the object classes (CBO) and lack of cohesion of methods (LCOM) were utilized, since, CBO and LCOM metrics can’t measure with the help of the automated tools. The result states that, none of these metrics is correlated with fault-proneness and a multivariate model doesn’t give any merits when distinguished with the lines of code metric. The software in our work was executed in C language and later this is developed in an object oriented system. This growth from C to object oriented system and a bad object-oriented design results in a correlation issues, which is explained above. Normally researchers want the correlation between fault-proneness and at least one metric. A module was the combination of a header and its corresponding file.

According to the General Regression Neural Network (GRNN) and Ward Neural Networks, software quality is estimated in (Thwin and Quah 2003). The class level metrics were utilized here and the report shows that GRNN gives much better performance when compared with the Ward networks. Evaluation parameters considered in our works were R², r, average square error, average absolute error, minimum absolute error, and maximum absolute error.

CART-LS, CART-LAD, S-PLUS, multiple linear regression, neural networks, and case based reasoning on a huge telecommunications system is enforced in (Khoshgoftaar and Seliya 2003). For this survey, 24 product and four execution metrics were independent variables. The two-way ANOVA randomized complete block design model is utilized as an
experimental design approach and multiple-pair wise comparison for performance ranking. Best performance was accomplished with CART-LAD algorithm and the worst one was S-PLUS. Additionally, they report that the principal component analysis doesn’t enhance the models performance, but it eliminates the correlation among the metrics and the model becomes more robust.

The multi-layer perception (Mahaweerawat et al. 2004) is utilized initially to recognize the fault-prone modules of 3000 C++ classes, which are gathered from various web pages and later, they were enforced by radial basis functions (RBF) to divide then in terms of fault types. By acquiring the knowledge C++ tool, metrics were gathered and the evaluation metrics considered here are accuracy, Type-I error, Type-II error, inspection, and achieved quality parameters. They decided to gather thousands of classes from various developers to construct a general prediction model, but prediction models can’t be modelled in a generic way and that model can work according to these 3000 classes. Accuracy, achieved quality, inspection, Type-I error, Type-II error were 90%, 91.53%, 59.55%, 5.32%, and 2.09% correspondingly. Also, the report states that they can’t recognize only 2.09% of faulty classes.

A constraint-based semi-supervised clustering scheme is suggested in (Seliya and Khoshgoftaar 2007), which make use of K-means clustering method to estimate the fault-proneness of program modules when the defect labels for modules were inaccessible. Nevertheless, their approach makes use of an expert’s domain knowledge to iteratively label the clusters, whether it is fault-prone or not. The expert labels clusters primarily base the author’s prediction and little statistical information. This means, our system doesn’t provide a well-established decision mechanism of the expert. Alternatively, the improvement of the dataset raises the clusters and iterations, which, in turn, it requires the expert to spend more time on these approaches.
The expert’s requirements avoid this method to be automatically proceeded and this give one main drawback in our system. Type-I error, Type-II error, and overall misclassification rate were considered as performance evaluation metrics. The report states that semi-supervised clustering schema gives good performance when compared with the conventional clustering methods and half of the modules can’t be labelled as noisy.

K-means and Neural-Gas clustering methods were utilized to cluster the modules in (Zhong et al. 2003) and then an expert who is 15 years experienced engineer, labeled every cluster as fault prone or not fault-prone by analyzing the statistical data like global mean, minimum, maximum, median, 75%, and 90% of every metric, along with the representative of every cluster. Mean squared error (MSE), average purity, and time parameters were utilized to compute the clustering quality. False positive rate (FPR), false negative rate (FNR), and overall misclassification rate parameters were utilized to compute the expert’s performance. Based on the opinion from experts, it is easy to label the modules with the help of Neural-Gas clustering. K-means clustering works much faster when compared with the Neural-Gas and Neural Gas’s overall misclassification rate’s performance is better when compared with the K-means clustering’s misclassification rate. Neural-Gas performed much better when distinguished with K-means based on the MSE parameter and its average purity was slightly good than K-means clustering's purity value.

An expert opinion and univariate linear regression analysis were utilized to examine the software faults with the help of method and class level metrics on two software systems which is established in Ericsson. For performance evaluation, accuracy parameter was utilized. Eleven experts were invited for this work and they foresee the fault-proneness of components. The report states that the statistical approaches are more successful when compared with the expert opinion and experts won’t foresee the faults much easily in the
huge datasets. The accuracy of prediction model works on the class level metrics and it results in a good performance, when distinguished with the model according to the component level metrics. While statistical approaches were inexpensive than the expert based approach, expert opinions assume the project specific problems at the time of prediction. The result states that the various experts select various components as fault-prone, and even if they select the similar components as fault-prone, they compute the diverse fault density values for components. Also, expert experience on products doesn’t affect out output.

2.2.5 Nearest Neighbor Techniques

The detailed literature review and existing trends in software fault prediction is explained in (Cagatay Catal 2011). Our work explains the metrics, methods, datasets and results of the earlier studies based on the software fault prediction. The issues that occur to estimate the software fault when earlier software fault data was missing and if it may be the new project. In order to rectify this issue, X-means clustering-based method is utilized, and also fuzzy clustering and k-means clustering-based methods were utilized in our work. But here metrics cost is expensive, so we require the influential model to foresee the faults with the restricted fault data. In our work, Naive Bayes algorithm is reported for semi-supervised fault prediction models with restricted datasets.

The fuzzy subtractive clustering method is enforced in [19], to estimate the faults count and they were utilized for module-order modelling, in order to divide the modules into faulty/non-faulty classes. Four models were established with 10 method-level metrics, additionally, they construct a model from process and product metrics. As a performance evaluation metrics, Type-I error, Type-II error and overall misclassification rate, effectiveness,
and efficiency were considered. The report states that, the process metrics doesn’t enhance the accuracy of the classification process and this doesn’t give acceptable results. So in our work, only faults were stated by customers and one module is determined as a set of source code files, which works altogether to do the function.

Fuzzy clustering is utilized initially in (Mahawerawat et al. 2002), and then, they were enforcing in applied radial basis function (RBF) to estimate software faults. Performance evaluation metrics considered here are Type-I error, Type-II error, overall misclassification rate, inspection, and completeness. Method level metrics like McCabe, and Henry and Kafura metrics, were utilized. RBF gives better results when compared with the multi-layer perception. Though the MLP’s completeness value is greater, its inspection cost was high too. RBF’s accuracy was 83% and MLP’s accuracy was 60%. Hence, RBF method was good, when distinguished with MLP for software fault prediction in our work.

25 projects of a telecommunication system and trained models on NASA MDP data were examined in (Turhan et al. 2009). The static call graph based ranking (CBGR) is utilized here and the nearest neighbour sampling constructs the defect predictors. The report states that at least 70% of faults can be recognized by examining only 6% of code with Naive Bayes model and 3% of code with CBGR model. Twenty nine static code metrics were utilized in our work and because of the deficiency of local fault data, the cross-company NASA data were utilized.

The object oriented software metrics on NASA’s KC1 dataset, foresee the high and low severity faults in (Zhou, Y., and Leung, H. 2006). They examine logistic regression, Naive Bayes, Random Forests, nearest neighbour with generalization techniques for fault prediction. Correctness, completeness,
and precision were considered as performance evaluation metrics here. The report states that low severity faults can be foreseen with a good performance than the high severity faults and number of children (NOC) isn’t considered for high severity faults prediction. They also states that rest of the Chidamber–Kemerer metrics assist for fault prediction.

2.3 SOFTWARE FAULT PREDICTION METRICS

A preliminary mapping study on software metrics is written in (B. Kitchenham 2010). Our survey was wider and it involves the theoretical and empirical studies, which were divided in the following criteria's: development, evaluation, analysis, framework, tool programs, and use and literature survey. Later, this study was narrowed to 15 studies evaluating metrics against fault proneness, effort and size. As a dependent variable in 9 out of 15 studies, the Fault proneness was utilized. The commonly utilized metrics were object-oriented (OO) metrics and, among these, CK metrics.

Given n methods M1, M2, ..., Mn contained in a class C1 which also contains a set of instance variables \{I_i\}. Then for any method M_i we can define the partitioned set of

\[ P = \{(I_i, I_j) | I_i \cap I_j = \emptyset\} \quad \text{and} \quad Q = \{(I_i, I_j) | I_i \cap I_j \neq \emptyset\} \]

then \[ \text{LCOM} = |P| - |Q|, \quad \text{if} \quad |P| > |Q| \]
\[ = 0 \quad \text{otherwise} \]

LCOM is a count of the number of method pairs whose similarity is zero.

The logical assessment of the software fault prediction is examined in (Catal, and Diri 2009). Later, a literature review on the same topic was written in (Catal 2011). They involve entire papers (focusing on empirical studies) regarding the software fault prediction. They divide the
studies in terms of metrics, methods and data sets. Metrics were divided into six types: method level (60%), class-level (24%), file-level (10%), process-level (4%), component-level (1%) and quantitative-level (1%).

A review that is similar to the Catal and Diri’s, but more comprehensive with respect to studies and analyses, was written in (Hall et al. 2011). Here, the paper involves on software fault prediction (concentrating once again on empirical studies). The primary target was to use context, independent variables and modelling techniques. A quantitative model across 19 studies was constructed to distinguish the metrics with respect -measure, precision and recall. Based on the quantitative model, OO metrics outperforms with the help of complexity metrics. The models using LOC with OO metrics also performs well and this is better than the model with complexity metrics. Models utilizing a combined range of metrics performed the best; while a model utilizes process metrics performed the worst.

It is confirmed that product metric, process metrics and object oriented metrics were broadly utilized in fault prediction techniques (Pradeep et al. 2015). But fault prediction result depends on human expertise separately from these metrics. For future work, we can expect the computation of the human expertise in software fault prediction techniques. The fault prediction totally depends on skewed data. But we have no confirmation of Fault prediction techniques for big data with real time and interactive data sets in this SR review, which is done as a future work.

For enhancing the quality of a system emphasize, the processes (Carol et al. 2014), minimizes the number of possible defects, but the quality measures and the techniques were enforced to enhance the quality can differ in effectiveness and the significance of the consequences of a defect and whether the measures and techniques were enforced to hardware or software. Desirable
experience occurs in computing the hardware quality. For instance: mean time among the failures is frequently utilized to compute the quality of a hardware component. Usually, long period failures are confirmed generally for hardware reliability. For computing the safety, the mean time among the failure isn’t enough. We required recognizing and moderating the defects which generate the dangerous conditions that affects the human life. Voting machine quality involves accurate tallies, but it also involves the mitigating design defects which facilitate interfere with the device. We have a fundamental consideration that a hardware device is appropriate over time. By minimizing the known problems enhances the hardware device quality. The failure distributions for hardware and software comparison doesn’t give reliability or security for a software component

Preethi et al (2011) propose few other metrics like: 1. Number of associated classes within a class (NAC): This metric gives the number of associated classes with a particular class. All kind of associations (e.g. association, aggregation and composition) will be used to count the number of associated classes. 2. Total Associated Class (TAC): This metric gives the number of times all associated attributes of a particular class type are used by methods of a user class.

The different researcher have been taken interest to construct a cross project defect prediction model with a number of metrics set like class level metric, process metrics, static code metrics but they could not build more feasible accurate models

Wan et al (2017) proposed and implement a manufacturing big data solution for active preventive maintenance in manufacturing environments. First, we provide the system architecture that is used for active preventive maintenance.
Singh et al (2017) performed a research on Genetic algorithms, they are identified to be one among the best means of solving a set of problems related to which very less information is given. Genetic algorithms are generalized algorithms and therefore they will function well in any kind of search space.

2.4 SUMMARY

In this chapter review of different research works which were brought in by different researchers for the fault prediction in the software is provided. The comprehensive description of the current research methods along with the performance measures were utilized in the work has been provided in order to give the elaborated view of the current research methods. The literature review of the conceptual models show that there are attempts to empirically prove theoretical concepts, but there is not any research that discusses the overall issues and their impacts to project performance. This study bridges this critical gap and this forms the foundation for the entire study.