Chapter 2

Literature Survey

2.1 Introduction

The main aim of software organization is to achieve project success and the project manager’s aim is to find out and fix defects (faults) in the early stages to achieve defect free software. Software project success depends on quality. Quality consists of several quality assurance activities such as testing, formal verification, inspection, fault tolerance and software defect (fault) prediction. Software development teams must deal with time, money, expertise and technology constraints to produce quality software.

The Standish Group’s 2011 CHAOS Report observed that more than half of software projects conducted between 2002 and 2010 were either described as challenged or complete failures; just 37 percent were classified as successful[33]. This survey presents the importance of evaluating the software primarily in its development phase and taking the essential safety measures before these effects come out.

Nowadays, many software development companies maintain their own software repositories. Surely it is helpful for the prediction of software defects. Literature reviews on the topics data mining, fuzzy logic, neural network and genetic algorithm reveal that each technique is used individually along with software engineering defect management.

Different kinds of software parameters and diverse feature reduction techniques are used in order to improve the performance of the models. However, these studies do not investigate the effect of dataset size, parameter set and feature selection techniques for software defect prediction. Software defect prediction using integrated techniques are advocated in this work. Software engineering researchers may lack expertise to adapt or develop integrated techniques or tools for mining while data mining researchers may lack the background to understand mining requirements in the software engineering field.

Software defect prediction using classification algorithms has been advocated by many researchers. However, several new literatures show the performance bottleneck by applying a single classifier in recent years. On the other hand, classifiers ensemble can effectively improve classification performance than a single classifier. The organization of this chapter is as follows. Introductory section provides an awareness of software defect prediction and its significance in the success of software projects. Section 2.2 provides research methodology. Section 2.3 provides the parameters that are necessary for error, fault
detection and prediction purposes. The approaches which are necessary to achieve this are also mentioned in this section. Section 2.4 provides the information needed for the identification of parameters for defect prediction using single classifiers in general approach. Section 2.5 identifies parameters for defect prediction using hybrid classifiers data mining, fuzzy logic, neural network and genetic algorithm approaches. Section 2.6 presents identification of several parameters applied for defect prediction using integration of sections 2.4 and 2.5. Section 2.7 is devoted to the discussion.

2.2 Research Methodology

In this research, we have explored the techniques used with mining software engineering defect data set with parameters which are identified and experimented using single classifier in general approach or in soft computing approach. This work throws light on classifier techniques and parameter identification in both the approaches which are identified through empirical investigations, carried out in various IT industries. The integration of general and soft computing approaches leads to ease the decision making capability of expert about the prediction of project’s success. Ultimately, this work integrates both the approaches for defect and fault prediction scheme is applied upon the empirical parameters. In order to achieve this entire research methodology consists of the following sequential steps.

a. Research perspectives in defect prediction, which also includes fault prediction.

b. Identification of parameters used in single classifier in general defect prediction approaches.

c. Identification of parameters for defect prediction purpose using hybrid classifiers in soft computational approaches

d. Identification of empirical parameters by integration of general and soft computational approaches for defect prediction

Parameter selection is one of the key steps in the process of software defect prediction due to the negative effect of irrelevant parameter on classification algorithms. Hence, selecting the most relevant and representative parameters is critical to the success of software defect prediction. Other major problems in software defect prediction are the availability of large number of classification models. Figure 2.1 infers the investigation of empirical parameters based on integration of general approaches (statistical, regression etc) and advanced approaches (data mining, fuzzy logic, neural network etc).
2.3 Research perspectives in defect and fault prediction

Software defects include both product and process defects such as defects at requirement analysis phase, design flaws, implementation errors and so on. Additionally, defects have varying degrees of complexity and severity as not all defects have same nature. Hence, to deliver defect free software, it is imperative to predict and fix all defects before product deployment to the customers, which is yet another key challenge of the software industry. However, software repositories have enormous information, which is highly beneficial in assessing the software quality. Hence, data mining techniques and machine learning algorithms can be applied on these repositories to extract the useful information on effective defect management [34].

The IEEE Standard Glossary of Software Engineering Terminology provides difference between Errors, faults, failures and defects [1]. Error: It is a human mistake, which produces an incorrect result. Fault: The manifestation of an error results in a software fault, which in turn results into a software failure. This translates into an inability of the system or
component to perform its required functions within specified requirements. Defect: A defect is considered the same as a fault although it is a term more common in hardware and systems engineering. A software defect is an error, flaw, mistake, failure, or fault in a computer program or system that produces an incorrect or unexpected result, or causes it to behave in unintended ways. In this review, the term fault is associated with mistakes at coding level. These mistakes are found during testing at unit and system levels. Although the anomalies reported during system testing can be termed as failures and remain persistent with using the term fault since it is expected that all the reported anomalies are tracked down to the coding level. In other words, the faults are considered as pre-release faults, which is an approach similar to the one taken by Fenton and Olsson in [35].

Software quality estimation consider defect acceptance, testing, authentication etc. Information belong to these factors helps to build a prediction method [36][37]. It consists of two steps, one is training and second is prediction. In the first case prediction model is designed with earlier metrics and defect data belonging to each software module. In the subsequent second step it is used to predict the defect prone class in new version. A software system with a complex code change, process is undesirable, since it will likely to produce system, which has many faults, and the project will face delays [38]. Also it include things like experience levels of the developers, the amount of time that the module spent in review, the number of defects found in reviews, the number of test cases (and unique test cases) run that touched the module.

Several public data sets include PROMISE software engineering repository data set which is made publicly available in order to encourage repeatable, verifiable, refutable, and/or improvable predictive models of software engineering [39]. However, a current literature work shown around 30% of defect prediction papers have used public data [40]. From this work, it is apparent that faults are, also considered as the one of the data type for defect prediction since there is a dependency between error, fault and defect. Further infers application of statistical model for fault prediction.

Author in [41] introduced the concept occurrence of defects are associated with complexity. The code change process in lieu of code further influences fault prediction, which is proven using complexity metrics by the author. From the above study, it is apparent that faults have intrinsic nature of inter dependency. However, the previous research studies have indicated the existence of dependency between the error, fault and defect. Thus, defects also will occur in clusters and not in isolation. This knowledge of defects appearing in clusters is beneficial when one proceeds towards defect management strategies such as defect
detection, elimination, prevention and prediction. It may be worth to recall back among all strategies defect prediction is more effective to yield or return on investment (ROI) in any company.

2.4 Identification of parameters for defect and fault prediction by single classifiers in general approaches

Software defects plays important role to take the decision about when testing should stop. Software defects are one of the major factors that can decide the time of software delivery. Along with the number of defects, the type of defect as well, the criticality of a software defect affects the software quality. Hence, all the software quality estimation approaches consider software defects as a parameter to estimate software quality. Defect prediction modelling has become a popular method for the early identification of defect prone code. It is now the topic of a large body of software engineering literature. This section depicts several defect prediction approaches in addition to the type of data used in various researches for the purpose of defect prediction.

In these approaches, complexity metrics on the code change process instead of the code [42][43]. Halstead, McCabe, lines of code and other metrics are being considered as parameters for defect prediction. Authors in [43] have shown that the number of lines added or removed in the past is a good predictor for future defects at the module level. This study defines defect prone as one which exceeds a threshold of debug code churn is defined as the number of lines added or altered due to error fix. However, this study has quantified defect proneness based on lines of code. Lines of code can be thought of as one of the data types to be looked into while performing defect prediction.

Authors in[44] infers the difficulty in identifying exact root cause of the defect. This may be due to error, fault and failure. This work further emphasises on the choice of relative code than absolute code churn indicating the need for application of fuzzy inferences towards accurate defect estimation and prediction.

Authors in [45] have used parameters (including code churn, past bugs and refactoring, number of authors, file size, age etc.) to predict the presence or absence of bugs in files of Eclipse. They have selected one set from each product related, process related software metrics and have used as defect or defect-free. The above technique indicates that churn of code metrics measured in terms of size and control flow of the source code can be used as one of the defect prediction parameters.
Authors in [46] have mentioned top ten list approaches and it has been validated on six open-source case studies: FreeBSD, NetBSD, OpenBSD, KDE, KOffice, and PostgreSQL. Their observation indicates that defect detection can be carried out by considering modifications of the code as one of the parameters. Authors in [47] predictions have been found to be quite accurate. This study indicates that defect predictions can be done based on previous history and existing code of the file using statistical approach.

Authors in [48] have used Bugzilla repositories of the Eclipse project, the authors have collected defect features and reported issues within few months. Subsequently, they used these extracted details about data to predict both the position of defect (the classes in which defects occur) and the number of reported defects in the coming month of the project. The system analyses software defect with respect to software criticality and its integration with software module. Additionally, works of several authors ([49]) indicate that they have prepared general framework for software defect prediction, the framework consisting of scheme evaluation and defect prediction components. These two classification patterns are single-trapezoidal-like curves and multiple-trapezoidal-like curves. This study indicates that the evaluation schemes are considered for defect prediction. Authors in [50] have developed necessary and sufficient conditions for software defect curves of the Goel-Okumoto NHPP model. This study indicates that the shapes of curves reveal the impact of defects. Importance of this study lies in the fact that by observing the number of remaining defects count during the testing process, it is possible to provide useful information on the software reliability. Without a realistic objective of software project plan, the effective development of software process is not possible. However in practice, more complex nonlinear relationships exist between metrics [50].

Authors in [51] have presented a metric based multi-agent system namely software project planning associate (SPPA). This assists managers in understanding and visualizing software project management (SPM) process defined in a software project plan. This study infers that empirical parameters are important for project implementation. Existing approaches use specific analytic forms typically linear for modelling correlations. From the above mentioned studies, we have been able to identify the various parameters such as lines of code (LOC), change code, change log, bugs and refactoring, file size, history of pre and post release code, churn of code metrics measured in terms of size and control flow of the source code, Chidamber and Kemmerer (CK) metrics and McCabe’s cyclomatic complexity and so on. These are the parameters used by various authors in general approach for defect
prediction model. Table 2.1 shows the list of single classifiers in general approaches and its parameters which influence the effectiveness of defect and fault prediction.

Table 2.1 Parameters list of defect/fault prediction in general approach

<table>
<thead>
<tr>
<th>General approaches</th>
<th>Defect/fault prediction Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>Size, effort, independent variable</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>McCabe’s cyclomatic complexity</td>
</tr>
<tr>
<td>Linear regression</td>
<td>CK metric suite: WMC, DIT, RFC, NOC, CBO, LCOM, LCOMN, LOC</td>
</tr>
<tr>
<td>Decision tree</td>
<td>PCA like Defect data CM1, KC1, MC1, PC1 and PC3</td>
</tr>
<tr>
<td>Descriptive stats</td>
<td>Change code, change log, file size, history of pre and post release code</td>
</tr>
<tr>
<td>Bayesian regression</td>
<td>Churn of code metrics, size and control flow of the source code</td>
</tr>
</tbody>
</table>

The work carried out by various authors reveals that several parameters are used in different approaches for defect / fault prediction. T. M. Khoshgoftaar et al ([44]) classification considers line of code parameter for linear regression. Nachiappan Nagappan et al ([45]) and Allen P. Nikora et al ([52]) states relative code churn is a better predictor than absolute churn in Bayesian regression approaches. Raimund Moser et al ([46]) have inferred the difficulty in identifying exact root cause of the defect. Abraham Bernstein et al ([48]) used eclipse for validation whereas Chenggang Bai et al ([50]) used shapes of curve for defect impact. These general approaches strategies have lead to a new defect / fault classification method based on an intelligent approach named ODC (Orthogonal Defect Classification). These indicate that when viewed from various perspectives of approaches, different parameters are deemed apt by the researchers and there is no standard baseline developed yet for an approach with standard set of parameters.

2.5 Identification of parameters for defect prediction using hybrid classifiers in soft computational approaches

Soft computing refers to a grouping of methodologies centred on Fuzzy Logic (FL), Artificial Neural Networks (ANN) and Genetic Algorithm (GA). They are flexible in processing the information and in handling real life ambiguous situations. Soft computing aims to exploit the tolerance for imprecision, uncertainty, approximate reasoning and partial truth in order to achieve tractability, robustness and low-cost solutions. The guiding principle is to devise
methods of computation that lead to an acceptable solution at low cost by seeking an approximate solution to a precisely or imprecisely formulated problem. It’s important to note that these methodologies are complementary and synergistic rather than competitive [29], [117].

To deliver defect-free software, it is imperative to predict and fix as many defects as possible before the product is delivered to the customer. Data mining techniques and machine learning algorithms can be applied on these repositories to extract the useful information. Some of these methods are used to determine the features that are closely correlative with the class attribute using association rule mining method. It is important to mention here that feature selection using data mining can highly improve the accuracy and efficiency of the software defect prediction model.

Clustering is considered to be one of the most significant techniques of data mining which is applicable in various domains such as life sciences, medical sciences, engineering and so on. Clustering technique can be viewed in various perspectives depending on operational environment which includes unsupervised learning in pattern recognition, numerical taxonomy in biology, typology in social sciences and partition in graph theory etc [53]. However, software is one of the domains which have laid its strong foot in almost all domains of applications [53].

One of the vital characteristics of the clustering process is to categorize the software projects into sensible groups. It facilitates in discovering similarities and differences upon which useful conclusions can be drawn from the mined information which in turn improves the quality of the software. However, quality is not a stable state but can vary depending on applications, nature of projects, industrial environment and so on. Hence, the choice of clustering algorithms should be based on availability of type of input data, purpose and application domain of the project. However, software is one of the domains which have laid its strong foot in almost all domains of applications.

The most habitually discussed hitch in any software organization is defect prediction to enhance quality and reliability [54]. To accomplish this objective, organizations are now integrating data mining techniques with software engineering processes effectively which includes efficient defect management. Although, residual defects number is not known and has to be identified and estimate it. For that reason, it is imperative to learn the trend of the left over software defect estimation and association rule mining based methods to predict defect associations and defect correction efforts [55]. The best size of attribute subset for building a prediction model will be very useful in establishing software defect prediction
Based on the properties, defects can be formulated together and then it will be feasible to calculate similar defects in other parts of the code written. Skewness in defect prediction datasets habitually referred to as learning from imbalanced datasets. However, it is uncertain what stratification techniques are most effective both generally and specifically in software defect prediction [56]. The problem of estimating defects in high-level software modules are investigated with a multi-instance learning (MIL) perspective in [57]. Arrangement of classification models is built using these frequent machine learners: logistic regression, naive Bays and decision trees. Cost-sensitive classification has been found to be very successful while considering different costs for prediction errors: > 75% of correctly classified files, a recall of > 80% and a false positive rate of < 30%.

A recent advance in software defect prediction has been widely debated, as merits of McCabe’s versus Halstead versus lines of code counts for generating defect predictors. Multi-Layer Perceptron (MLP) networks were trained using two-thirds of the 54 model development observations for training and one-third for a testing set. Training stopped when the testing error was minimized and the lowest testing error was also used to select the particular network architecture. Neural network is one of the best artificial intelligence techniques to provide supervision to a classification technique (unsupervised technique). Novel neural networks (NNs) architecture and algorithms are available for predicting defects using preference parameters [58]. This work is useful while selecting important parameters of software projects.

Fuzzy Logic is an effective soft-computing technique to solve uncertainties due to imprecise inputs for generating linguistic outputs. Fuzzy logic model uses the concepts introduced by Lofti A. Zadeh [24]. Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is approximate rather than precise. A fuzzy set expresses the degree to which an element belongs to a set. The characteristic function of a fuzzy set is allowed to have values between 0 and 1 which denotes the degree of membership of an element in a given set. They have been employed in various real life applications. Fuzzy logic modelling techniques such as Fuzzy C-means clustering (FCM), fuzzy inference etc have been shown to be useful additions to the existing statistical and machine learning techniques used for modelling software development [59].

Fuzzy Logic has gained popularity in recent times as a sensible technique to achieve improved estimation accuracy of variables in any process. Fuzzy C-means clustering (FCM) is one of the most recent contributions to the field of Artificial Intelligence (AI) and data
clustering. The variables range from software resource estimation to resource allocation.

A significant motivation for using fuzzy logic lies in its ability to estimate required effort much earlier in the development process. Since many of the independent variables in software metric models are either difficult to quantify (for e.g. complexity) or are only known to a rough degree (such as system size), the use of fuzzy variables seems intuitively appealing. The basic underlying assumption is that project managers are able to classify systems using fuzzy variables with reasonable levels of both accuracy and consistency.

The case study in [12] is based on actual project data. Project effort, project duration, levels of experience with equipment, levels of experience in project management, number of basic transactions, number of data entities, raw and adjusted function point counts are all standard methods for counting the functional size of a system. Management of metrics in a software system is a complex task. Estimation of defect-proneness in object-oriented systems at design level is developed using a novel methodology in which models of relationship between CK metrics and defect-proneness index can be achieved [57].

A project manager is a professional in the field of project management. Project managers have the responsibility for planning, execution and closing any project typically relating to construction industry, engineering, architecture, computing and telecommunications. Maturity and competence of a project manager in efficient prediction and estimation of resource capabilities are one of the strategic driving forces towards the generation of high quality software [39]. The use of artificial intelligence and intelligent techniques provide supervision to the existing unsupervised approaches like clustering and other statistical approaches. Artificial intelligence system is initially trained and is made ready for taking suitable decisions subsequently.

Authors in [54][56] have proposed an ideal software defect management and prediction system. They have carried out research and have analysed several software defect prediction methods based on data mining techniques and specific models. This empirical study helps managers in achieving the defect prediction objective.

The defect parameters identified in data mining, fuzzy logic, neural network and genetic algorithm are NASA data sets such as CM1, KC1, MC1, PC1 and PC3. Weibull parameters, software history metrics, churn and static code metrics, schedule, effort and quality are used in fuzzy c means (FCM) clustering techniques. The parameters used in neural network are developer metrics such as the number of code churns made by each developer, number of commitments made by each developer, the number of developers and
the number of developers for each module. One also has to consider the human factors for improving software reliability. Table 2.2 indicates the list of defect and fault prediction parameters that influence the effectiveness of defect and fault prediction by hybrid classifiers in soft computing approaches.

Table 2.2 Parameters list identified for Soft computing approach of defect/fault prediction

<table>
<thead>
<tr>
<th>Soft computing approaches</th>
<th>Defect/fault prediction Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data mining(DM)</td>
<td>Software repositories.</td>
</tr>
<tr>
<td>Fuzzy C means clustering(FCM)</td>
<td>software history metrics, churn and static code metrics, schedule, effort and quality</td>
</tr>
<tr>
<td>Neural Network(NN)</td>
<td>Number of code churns made by each developer, number of commitments made by each developer</td>
</tr>
<tr>
<td>Genetic Algorithm(GA)</td>
<td>Also consider the human factors for improving the software reliability, counts the number of developers and number of developers for each module</td>
</tr>
</tbody>
</table>

2.6 Identification of empirical parameters using integration of general approaches and soft computational approaches

The identified list of defect / fault prediction parameters are obtained from general and advanced approaches. Despite the prevalence of very interesting research work, there still exists a huge avenue for research to be carried out on empirical data. Researchers in this field empirically explore a range of software engineering questions using software repository data as the primary source of information. Some commonly explored areas include software evolution, models of software development processes, characterization of developers and their activities, prediction of future software qualities, use of machine learning techniques on software project data, software bug prediction, analysis of software change patterns and analysis of code duplicate. There has also been a stream of work on tools for mining software repositories and techniques for visualizing software repository data.

Lessmann et al., 2008 [61] investigated the performance of classification algorithm. To compare with different classifiers the software defect prediction, experiments were conducted using 10 public domain datasets from NASA Metric Data repository using 22 classifiers. The general impression is that the predictive accuracy metric based classification is useful. The results also indicate that the importance attached to particular classification algorithms is not significant as generally assumed. The results show that there is no significant difference among the top 17 classifiers. Munson et al., 1992 [62] investigated
statistical based methods such as discriminant analysis for the detection of fault prone programs. In this paper, it was proposed to implement principal components to reduce multicollinear complexity metrics to uncorrelated measures on orthogonal complexity domains. The transformed data was used to classify the programs. Eleven software metrics were computed from the programs and data prepared for the classification engine. The misclassification rate was 10% which shows a high degree of classification. Mitchell, T.M., in 1997[64] attempted to classify data using Naïve Bayesian algorithm. Naïve Bayes is one of the popularly used learning algorithms in data mining and machine learning. It is popular because it is an efficient and effective inductive learning algorithm. Classification based on Naïve Bayes algorithm gives very competitive performance due to its conditional independence assumption.

Ohlsson et al., [63] derived metrics using design documents of telecommunication software modules to predict fault prone modules prior to testing with accurate results. Menzies et al., 2004 [43] have proposed Naïve Bayes learners for studying the defect detectors from static code measures. Comparison of Naïve Bayes learners and entropy-based decision tree learner is done to show the effectiveness of Naïve Bayes learners. The study concludes that accuracy is not an effective way to assess those detectors. When using Naïve Bayes learners on heavily stratified data, 200-300 examples are enough to learn adequate detectors. Riquelme et al., 2009 [26] used the promise repository to obtain the software metrics program dataset. They have also proposed a genetic algorithm search for rules characterizing subgroups with a high probability of being defective. The genetic algorithm handles the problem of unbalanced datasets efficiently especially when the unbalanced sets consist of more non-defective samples than defective ones.

The issues affecting defect prediction can be summarized as follows: The unknown correlation linking defects and failures, false claims about software decomposition. Evolutionary sampling which is a genetic algorithm based data sampling method is used to improve software quality modelling for high-assurance systems. The proposed sampling is compared with the existing data sampling techniques. Two case studies of software quality model both before and after applying the evolutionary sampling technique has been presented. The improvement in the performance of software quality models by using evolutionary sampling is shown empirically.

In [26] proposed change classification method for predicting dormant software bugs. Change classification is based on machine learning classifiers which help in determining the similarity of change to previous buggy changes or clean changes. The presence of bugs is
predicted using the change classification. The classifier is trained using the features from revision history of the software. Thus the classification of the changes in the software as buggy or clean is achieved. The results show 78% accuracy. The change classification is superior since it has small prediction granularity. Also semantic information of source code is not required for classification. Change classification works on a broad array of programming languages. The work carried out by above researches for defect / fault prediction using integrated strategy of soft computing and general approaches can be placed under the perception of binary classification approaches. Binary classification involves simple process metrics and metrics which are even more advanced such as source code and entropy of source code metrics.

Yi Liu et al., 2010 [26] investigated the problem of software quality classification modelling using the history of metric dataset obtained from single software project. The classification modelling obtained from a single dataset is generally not sufficient to build a strong and precise model. To deal with the issue, software quality classification modelling has been done using multiple datasets sourced from different software projects. Subsequent study shows that multiple datasets are used for validation can realize strong genetic programming based models. The effectiveness of the modelling using multiple datasets is extensively studied in this paper.

A novel general approach based classifier consisting of training, multiple-dataset validation and voting phases have been proposed. The datasets used for experimentation were obtained from NASA software projects. The performance by the proposed classifier was compared with the results of seventeen other data mining techniques. The comparison shows that the proposed approach is more effective and accurate with the use of multiple datasets.

Song et al., 2006 [54] have proposed prediction of defect associations and defect correction effort based on association rule mining methods. Test resources are more effectively allocated for detecting software defects. The proposed method has been applied to more than 200 projects. The experimental results show that the accuracy achieved is high for both defect association prediction and defect correction effort predictions. The result of the proposed method was also compared with PART, C4.5 AND Naïve Bayes methods. The comparison shows that the proposed method accuracy is higher by at least 23 percent.

Attarzadeh, I. [65] has incorporates Constructive Cost Model (COCOMO) and ANN-COCOMO II for providing more accurate software effort estimates in the early phase of software development.
Xin Jin et al., 2006 [22] attempted to provide software reliability for Software Engineering Management. They have come out with a list of metrics which have been implemented in a common dataset. This work attempts to improve all the measures made by them by incorporating additional metrics with a combined effort. Good results have been obtained from experimenting with the artificial immune recognition system of classifiers. Menzies et al. [42] have taken effort into account in their defect prediction approach. Their observation is that a few prediction algorithms are outperformed by manual methods while using static code metric data. The time to require a potential bug files to be discovered, a shorter file involves less effort [66].

Authors in [39] have shown that if a trivial defect prediction model which predicts the presence of bugs in large files performs well with a classic evaluation metric such as the ROC(Receiver Operator characteristic Curve) curve. They have evaluated two effort-aware models and have compared them with a classical prediction model [39]. Authors in [67] revisited common findings in defect prediction while using effort-aware performance measurements.

Yuan, et al [120] explains a method to calculate the defect prediction, reliability using fuzzy neural network. In order to improve the accuracy further it is uses adaptive neuro fuzzy inference system. Ebru Ardl et al., 2009 [68] investigated the modules of feed forward neural network. Feed forward neural network is the first and the simplest type of artificial neural network. Since faults are mainly found in the modules of neural network, the study investigates the most severely affected modules in comparison with other modules. Jianhong, et al (2010) has explored five neural network based techniques and comparative analysis has been performed for severity of faults present in task based software systems. Out of five neural networks based techniques, resilient back propagation algorithm based neural network has been found to be the best for modelling of the software components into different levels of severity of the faults. Which have major faults and which require immediate attention?

Catal et al., 2009 [32] modelled Artificial Immune System based on the Human immune system for defect prediction. The proposed classifier imitates the behaviour of the antigen and the antibody during an attack by pathogens into the human biological system. The evolution of the immune system to new attacks is modelled to solve the software defect prediction problem. The approach of Achcar et al., 1991 [32] to the software reliability prediction extends the Bayesian approach by using Poisson distribution to propose a novel software reliability model. The Bayesian inference models used Metropolis-within-Gibbs algorithms for the Moranda’s model. Model selection was based on the predictive density.
Anderson et al., 2007 [32] and Hassan et al [119] proposed a novel method for analysis of fault distributions in software systems. The method used replicated quantitative analysis which was mathematically modelled and proved in detection of fault distributed across complex software system.

Basili et al., 1996 [69], [118] extensively investigated design metrics for object oriented software with the understanding that errors multiply at each stage of the software design process. The idea is to identify metrics at the design stage so that prediction can be done earlier to remove defects. The proposed method was validated with good results. Emam et al., 2001 [32] proposed enhancement in the computation of design metrics based on the previous work on predicting faulty modules using object oriented design metrics. The proposed method of feature extraction performed better than the previous models. The disadvantages of the proposed method were also highlighted.

Selby 1990 [46] successfully implemented software metrics for predicting software module reuse. Comparison of development variable in various modules was done through nonparametric statistical models. Factors which contribute to the successful reuse of software are the module implementation factors and the module design factors. Cartwright et al., 2000 [33] investigated an industrial object-oriented (OO) system made of 133,000 lines of C++ using empirical methods. The data system was a subsystem of a telecommunication product.

The study shows that OO constructs such as inheritance, polymorphism are not useful. Also classes in inheritance structures are three times more defect prone than the classes that are not in inheritance structures. Prediction systems were constructed using a number of states and events per class. Though the prediction systems have only local significance, the need of suites of metrics in OO technology is not required; thus, measurement technology becomes more accessible.

Liu et al., 2010 [29] proposed a novel genetic programming based on search approach for software quality modelling with multiple software project repositories. The training on multiple projects on software quality method, which can effectively sum up the quality trends of the development organization. This approach includes three strategies provides a clear direction for organization’s software measurement data repositories for improving software quality modelling.

The work carried out by above researchers for defect/fault prediction using integrated strategy of soft computing and general approaches can be placed under the perception of ranking based on defect / fault prediction parameters. In [70] they have used several graph metrics including McCabe’s and CK metrics in conjunction with Briand’s coupling metrics to
predict defects. Based on data from NASA Metrics Data Program (MDP), the authors compared the impact of using metrics with the impact of using Naive Bayes, OneR, and J48 algorithms.

Design metrics play an important role in helping developers understand design aspects of software which will in turn help in improving software quality and developer productivity. In this work, author has provided empirical evidence supporting the role of object oriented design complexity metrics specifically a subset of the CK metrics suite [70] in determining software defects [71]. Authors in [72] have performed a similar analysis on Mozilla. Researchers have also used other metric suites in web and e-mail suite called Mozilla. Fenton and Neil (1999) argue that static code measures alone will not be able to predict software defects accurately. Lots of different software metrics are available in the literature which can be used for defect and fault prediction.

From the above studies, it is quite apparent that earlier studies included more of static code features. Consequently with the advancement of technology, it is realized that the effect of static code metric on defect and fault prediction in addition to various other influencing measures, process metrics also need to be investigated. Table 2.3 provides a list of parameters which enable defect and fault prediction. Table 3 thus indicates identified list of parameters as obtained from the integration of general and soft computing approaches that influence effectiveness of defect and fault prediction.

Table 2.3 Parameters list identified for integrated approach of defect/fault prediction

<table>
<thead>
<tr>
<th>Sl no</th>
<th>Defect/fault prediction Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Total project time in hours, Inspection time scheduled</td>
</tr>
<tr>
<td>2</td>
<td>Number of inspectors involved, Experience level of inspectors</td>
</tr>
<tr>
<td>3</td>
<td>Defect count estimation, Number of defects detected, Defects</td>
</tr>
<tr>
<td>4</td>
<td>Number of defects not captured, Defects due to bad fixes, Testing</td>
</tr>
<tr>
<td>5</td>
<td>Number of testers, Experience level of testers (years), Defect count</td>
</tr>
<tr>
<td>6</td>
<td>Number of defects detected</td>
</tr>
</tbody>
</table>
2.7 Discussion

This review work provides an insight into general and advanced approaches in addition to the integrated approaches of both with the intention of identifying all parameters for defect/fault prediction purposes as studied by various researchers. This comprehensive evaluation further indicates the integrated approach for identifying the list of parameters that will enable one to perform defect/fault prediction during software development process. Additionally, this integrated approach of general and soft computing approaches has paved the way to view the identified list of parameters as most eligible parameter list from the following three perspectives.

- Binary classification perspective of defect/fault prediction parameters.
- Effort aware ranking based on defect density.
- Ranking based on defects/fault prediction parameters.

The data set obtained by the authors for carrying out their research for the purpose of defect and fault prediction may not be considered as optimized list of parameters. Hence, research gate is always wide open for both industry and R&D personnel to plug in additional list of parameters in order to achieve more accuracy.

The review conducted so far has indicated a strong need to carry out research on empirical data. The above discussions can be taken up further to introduce novel approaches with well defined and optimized list of parameters that can be applied on empirical data set in such a way that it leads to formulation of parameter list like NASA data set, PROMISE data set etc. These theoretical and empirical evidences which demonstrate the methodology to combine the predictions of multiple classifiers will lead to more accurate decision-making.

2.8 Summary

Software defect prediction is vital for evaluating products in terms of software quality prior to the software implementation. Defect prediction concerns with the resource allocation problem having an accurate estimate of the distribution of bugs across components which help project managers to optimize the available resources by focusing on the problematic system parts. Different approaches are suggested by various researchers to predict future defects in software systems which vary in the data sources that they use, in the systems that they validate upon and in the evaluation technique employed.

This review article put forth many defect prediction approaches that are being carried out and implemented since a decade. These approaches are heterogeneous. A wide range of statistical
and machine learning approaches are used to build models. Work carried out by several researchers has indicated the use of publicly available NASA data for prediction purposes. The work further confirms that industrial data is also used for the above said purpose. However, research with industrial data is more preferable than the data set available in the free source in order to claim the effectiveness of prediction process. Additionally, from the survey of research, it is found that a wide variety of independent variables which are used for the prediction purpose can be categorized under static code metrics, change metrics and previous fault metrics. Independent variables such as lines of code, complexity metrics, process metrics, module size, age of file are strongly correlated to defects in some studies but have no correlation to defects in others.

This research analyses several techniques based on supervised and unsupervised learning and further indicates that recently evolved advanced intelligent approaches show better results when compared to the traditional classification approaches. However, there is no promising base line approach that results from the above studies. This is because the prediction indicators in terms of parameters and approaches that the researchers have used are different for different applications. However, it is worthwhile to study all these investigations which encourage developing a model using industrial data for effective software development through optimal defect and fault prediction techniques. It further calls for the need to introduce a standard baseline having a standard list of prediction parameters in compliance with a specific approach which is thereby applicable for a framework of applications.