A Review of Metrics and Modeling Techniques in Software Fault Prediction Model Development

Rinkaj Goyal[1],

Pravin Chandra[2],

Yogesh Singh[3]

[1] USICT, Guru Gobind Singh Indraprastha University, Sector 16C, Dwarka, Delhi-110078 [1]rinkajgoyal@gmail.com

[2] USICT, Guru Gobind Singh Indraprastha University, Sector 16C, Dwarka, Delhi-110078

[2] chandra.pravin@gmail.com

[3] USICT, Guru Gobind Singh Indraprastha University, Sector 16C, Dwarka, Delhi-110078[4]

[3]ys66@rediffmail.com

Abstract: This paper surveys different software fault predictions progressed through different data analytic techniques reported in the software engineering literature. This study split in three broad areas; (a) The description of software metrics suites reported and validated in the literature. (b) A brief outline of previous research published in the development of software fault prediction model based on various analytic techniques. This utilizes the taxonomy of analytic techniques while summarizing published research. (c) A review of the advantages of using the combination of metrics. Though, this area is comparatively new and needs more research efforts.

Keywords: Metrics Suite; Object oriented metrics; Software fault prediction; Software Metrics.

1. Introduction

Development of fault prediction models in software engineering is a field more than three decades old; however is still an emerging aspect of empirical software engineering (Catal and Diri, 2007; Kaner and Bond, 2004; Matsumoto et al., 2010; Radjenovic et al., 2013). The resurgence in this field occurs due to availability of public available data as repositories in recent decade and as well as due to the development of other numerical techniques, which have been researched in considerable depth(Dick et al., 2004).

A fault prediction model uses statistical methods to assess and quantify the relationship between different metrics and fault-proneness of a software module even before it is released (Catal and Diri, 2009; Catal et al., 2011; Hall et al., 2011; Raj Kiran and Ravi, 2008).

Different object-oriented metrics have been proposed in the literature due to the increased usage of object-oriented technology in software development(Aggarwal et al., 2009, 2006; Anh, 2010; Arisholm et al., 2010; Babic, 2012; Caglayan et al., 2010; Catal, 2011; Chowdhury and Zulkernine, 2011).

Predictive models quantitatively estimate some aspect of system quality and their efficiency is determined by fault history data and applied quality evaluation procedures(Corazza et al., 2010; Couto et al., 2012; Hong et al., 2010; Janes et al., 2006; Jones, 2008; Khoshgoftaar et al., 2006; Lavazza and Robiolo, 2010; Li and Henry, 1993; Li et al., 1991; Luo et al., 2010) Object oriented development needs a different strategy towards the development of metrics. Since, object-oriented technology utilize objects as its building blocks and contrasting from procedural systems, which use algorithms instead. The derivation and consequently the selection of appropriate metrics depend on the identification of attributes of objects and peculiarities of object-oriented software development process. These metrics not only indicate the complexity of an object and its association (interaction) with other objects, but also measure different characteristics of a quality model **(Figure .**Error! Reference source not found.**).**

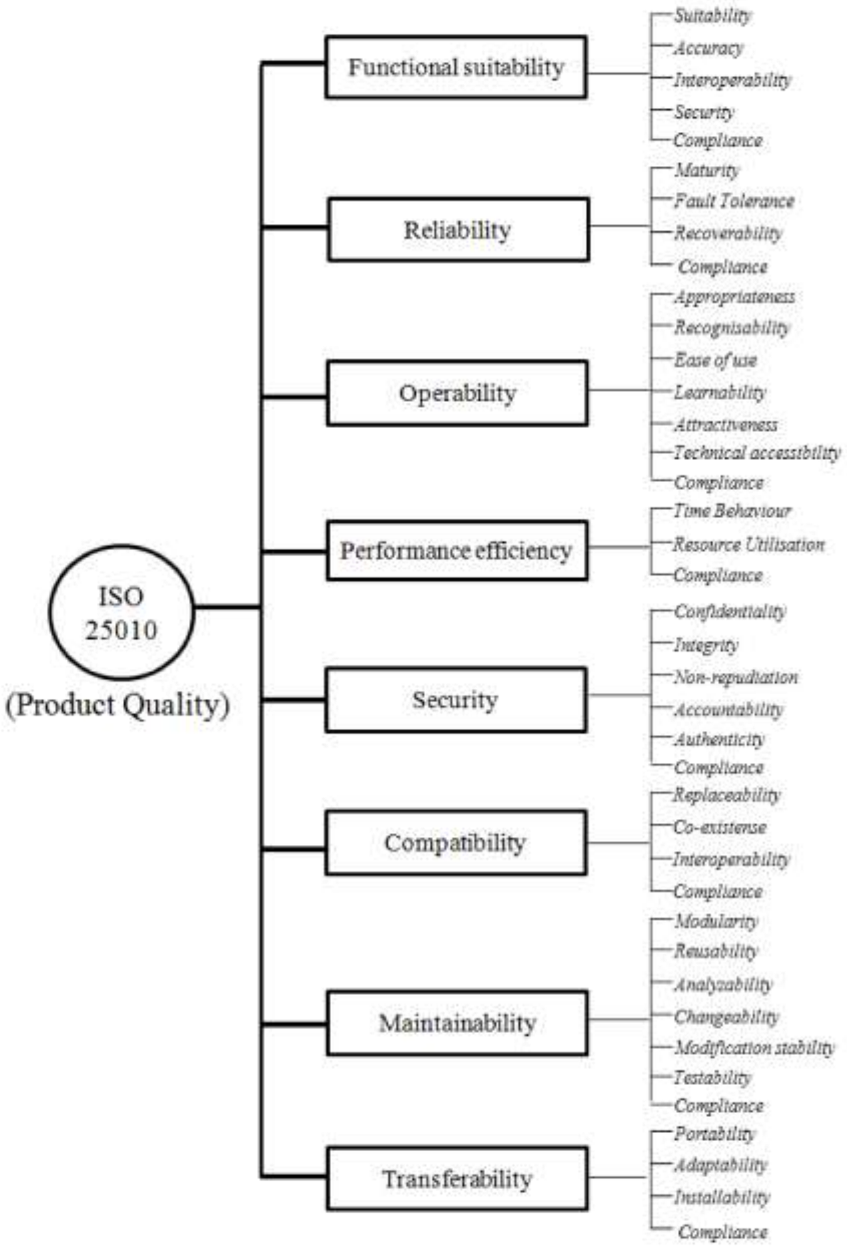

Figure 1: ISO/IEC 25010 Software Quality Standard (Adapted from Wagner et al. (Wagner, 2013)

2 Software Metrics and Suites: A Survey

Software metrics can be categorized as product metrics, process metrics or resource metrics. Product metrics measure different features of developed programs like Methods and Class level metrics in object-oriented systems. Process metrics are related to the measurement and quantification of activities like design, implementation, testing, and maintenance. Resource metrics focus on all other resources involved in development such as programmers, cost of the product and processes, etc. (Ebert and Dumke, 2007; Koru and Liu, 2005; Laird and Brennan, 2006; Lanubile and Visaggio, 1997).

These metrics have shown a corresponding relationship with a variety of external quality characteristics of software, such as reliability, testability and maintainability (Alshayeb and Li, 2003; Li and Henry, 1993; Mair and Shepperd, 2011).

Carapuça et al. (Carapucca and Others, 1994) suggested a classification skeleton that represents the taxonomy of Object oriented metrics. This framework is known as TAPROOT ((Taxonomy Précis for Object-Oriented Metrics) portrayed as a tubular arrangement with two independent vectors (Figure 2); different aspects of measurement (design, size, complexity, reuse, productivity, quality) and the granularity (method, class, system) of an object-oriented system. Though, there are no obvious boundaries between different categories and overlapping may be observed. However, this framework promotes the necessity of relevant metrics adequately to address a particular dimension of the software module. Figure 3 sketches the OO design measures apprehending varying dimensions and architectural quality of a class identified by Briand et al. (Briand et al., 1998). These measures associate to the fault-proneness of a class.

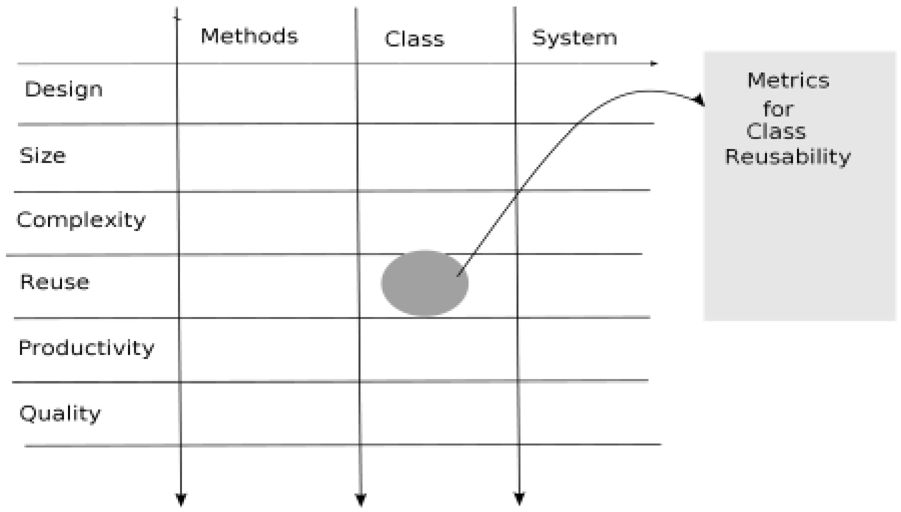

Figure 1: Taxonomy for Object-Oriented Metrics.

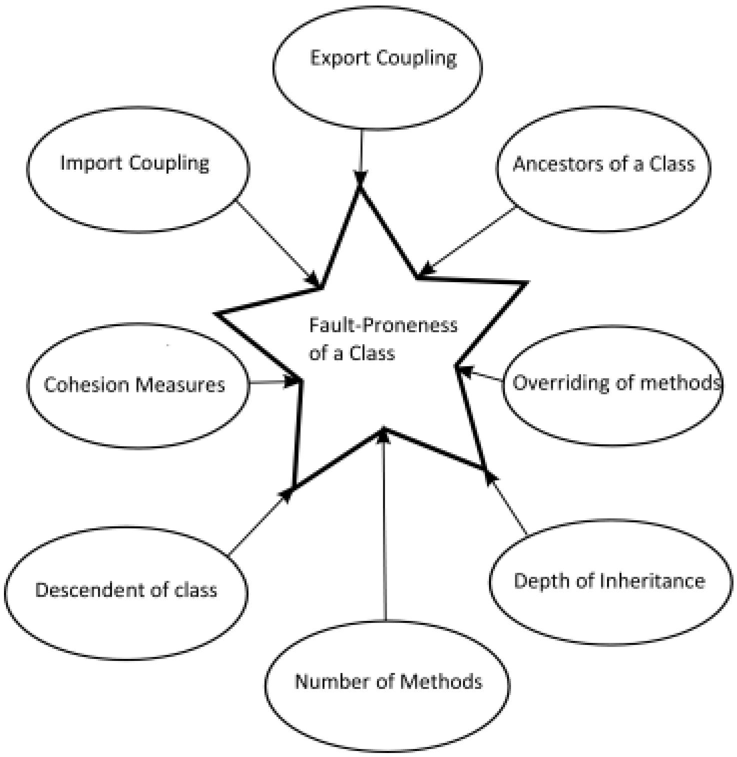

Figure 2: OO Design Measures Related to Fault-Proneness.

A concise summary of the development of different metrics (metrics suite) in the chronological order of their reporting is given below;

McCabe T. (McCabe, 1976) proposed a graph-theoretic measure to compute program's structural complexity known as Cyclomatic Complexity (CC). When a program is modelled as a control flow graph, CC is defined as follows:

$$CC=e-n+p \quad (1)$$

Where n = number of vertices; e = number of edges and p = connected components.

Conceptualising coupling as the critical complexity measure for fault prediction, Li, W. and Henry Li (Li and Henry, 1993) suggested two metrics Message Passing Coupling (MPC) and Data Abstraction Coupling (DAC), based on the coupling through message passing moreover, Abstract data types (ADT) declared in the class. Two size metrics i.e. SIZE 1 and SIZE 2 were also recommended, addressing the ambiguity in the determination of the size factor of an object-oriented program

(Table 1).

TABLE 1: METRICS PROPOSED BY LI AND HENRY (Li et al., 1991)

Metric Name	Description	Category
MPC(Message Passing Coupling)	Number of send statements defined in a class	Methods Design
Number of Methods (NOM)	Number of local methods	Method Complexity
SIZE1	Number of semicolons in a class	Attribute size
SIZE2	Number of attributes + Number of local methods	Attribute size

Chidamber, S. R. and Kemerer (Chidamber and Kemerer, 1994) proposed an extensively applied and validated metrics suite, commonly identified as Chidamber & Kemerer (CK) metrics suite with six metrics (Table 2).

Hitz, M. and Montazeri (Hitz and Montazeri, 1995) discussed flaws in the determination of coupling constituent in CK metrics suite. They proposed two coupling based metrics, Coupling among objects(CLO) and Coupling among classes (CLC) by analysing the coupling between classes and object as two distinct impressions (Table 3).

TABLE 2: METRICS PROPOSED BY CHIDAMBER & KEMERER (Chidamber and Kemerer, 1994)

CK Metric	Description	Category
Coupling Between Object classes (CBO)	represents the dependence of one class over other classes	System Complexity
Depth of the Inheritance Tree (DIT)	represents the length of the longest path from a given class to the root class in the inheritance tree	Class Design
Lack of Cohesion Metric (LCOM)	represents the count of method pairs in a class with zero similarities	Class Design/Method Complexity
Response for the classes (RFC)	represents the sum of the number of local	Class Design

	methods and remote methods	
Weighted Methods per Class (WMC)	represents the sum of the complexity of methods.	Method Complexity
Number of Children (NOC)	represents the count of the number of immediate subclasses of a class.	Class Complexity

TABLE 3:METRICS PROPOSED BY HITZ & MONTAZERI

Metric Name	Description	Category
CLO(Coupling among objects)	Represents dynamic dependencies between objects	System complexity
CLC(Coupling among classes)	Represents static dependencies between implementations	System complexity

Tegarden et al. (Tegarden et al., 1995) introduced following metrics through verifying that interaction and inheritance are the determining factors in the coupling aspect of a class. Whereas, features like association and generalization-specialization contributes towards the cohesiveness (Table 4).

Abreu et al. (e Abreu and Melo, 1996) proposed a MOOD (Metrics for Object Oriented Design) metrics suite comprising of the metrics listed in Table 5. These metrics capture core architectural ingredients of an object-oriented program like encapsulation, inheritance, polymorphism and message passing. Bansiya et al. (Bansiya and Davis, 2002) proposed QMOOD (Quality model for object-oriented design) metrics suite with an assessment of total quality index as super metric. These eleven metrics are based on the design quality attributes defined in ISO 9126 and possess an edge of early computability in the design process (Table 5).

TABLE 4: METRICS PROPOSED BY TEGARDEN, D. P et. al. (Tegarden et al., 1995)

Metric Name	Description	Category
CLD(Class-to-leaf depth)	Count the maximum levels that are below the class in the inheritance hierarchy	Class complexity
NOA(Number of ancestors)	Count of the parent classes of the class.	Class complexity
NOD(Number of descendants)	Count of the descendent classes of a class.	Class reusability

TABLE 4: METRICS PROPOSED BY ABREU et. al. (Abreu and Melo, 1996)

Metric Name	Description	Category
Method Hiding Factor (MHF)	Average (in per cent) of the methods visibility.	Class design
Attribute Hiding Factor (AHF)	Average (in per cent) of the attribute visibility.	Class design
Method Inheritance Factor (MIF)	Average (in per cent) of methods reusability.	Method reusability
Attribute Inheritance Factor (AIF)	Average (in per cent) of attributes reusability.	Class reusability
Coupling Factor (COF)	Average (in per cent) of class coupling.	Class complexity
Polymorphism Factor (POF)	Average (in per cent) of methods overridden.	Method complexity

Software measurement research community is actively involved in identifying new OO metrics addressing more quality attributes of Object-oriented software. Recent work in this regards includes the following;

Michura et al. (Michura et al., 2013) proposed complexity metrics to determine the difficulty in implementing changes through the measurement of a method's complexity, diversity, and complexity density (Table 6).

Wang et al.(Wang and Shao, 2003) proposed Cognitive complexity as a new measure to determine the complexity by taking the cognitive and psychological parameters into account.

These parameters consider internal structures of the artifact along with the processed input-output into consideration to measure particular facet of the quality of a software. Misra et al. (Misra and Adewumi, 2014; Misra, 2011; Misra et al., 2012)proposed following cognition driven complexity measures (Table 7).

TABLE 6:METRICS PROPOSED BY MICHURA et al. (Michura et al., 2013)

Metric Name	Description
Mean Method Complexity (MMC)	Measures the complexity of a class method obtained by dividing method's cyclomatic complexity with the number of methods in a class.
Standard Deviation Method Complexity (SDMC)	measure the method diversity of a class by taking the deviation of a methods complexity from the mean of methods complexity into consideration.
Proportion of Nontrivial Complexity (PNC);	measures the class complexity density by identifying the proportion of methods whose complexity is not one.

TABLE 5: METRICS PROPOSED BY MISRA et al. (Misra et al., 2012)

Metric Name	Description
Method Complexity (MC)	Measures the complexity of a method by taking logical structures used in a method into consideration. This metric is computed by assigning a weight to each logical structure involved in the implementation of a method followed by summing up the complexity thus obtained for all methods of a class.
Coupling weight for a class (CWC)	Measures the coupling effect between classes by not only considering the number of messaged passed, but also taking the complexity of calling and called functions into consideration.
Attribute Complexity (AC)	Measures the complexity induced in the class due to data members of a class. This metric is obtained by summing up the number of attributes.
Weighted Class Complexity (WCC)	Measures the class complexity as a whole by summing up the methods and attributes complexity.
Code Complexity (CC)	Measures the complexity introduced due to inheritance by differentiating between the influence of sibling and child-parent relationship in the determination of the overall impact.

3 Review of Modeling Techniques

In recent years empirical software engineering has seen an increased usage of various data analytic techniques accruing to the public availability of a multitude of software repositories (Harrison et al., 1998; Mende, 2010; Menzies et al., 2010; Mertik et al., 2006; Perry et al., 2000; Rodriguez et al., 2012; Runeson et al., 2006; Seaman, 1999; Shepard et al., 2001) and progressive research shown by machine learning and data mining community. Nonparametric techniques like Regression Tree, Random Forest, Support Vector Machine, Neural Network etc. have been extensively reported tang(Brady and Menzies, 2010; Lessmann et al., 2008; Malhotra et al., 2010; Succi et al., 2003; Tang et al., 1999; Tichy, 1998). Following is the review of fault prediction model's evolution based on the grounds of applied data analysis routines. Fig. 4 outlines the categorization of the analysis methods studied in this section along with their position in the hierarchy of the broad spectrum of data science.

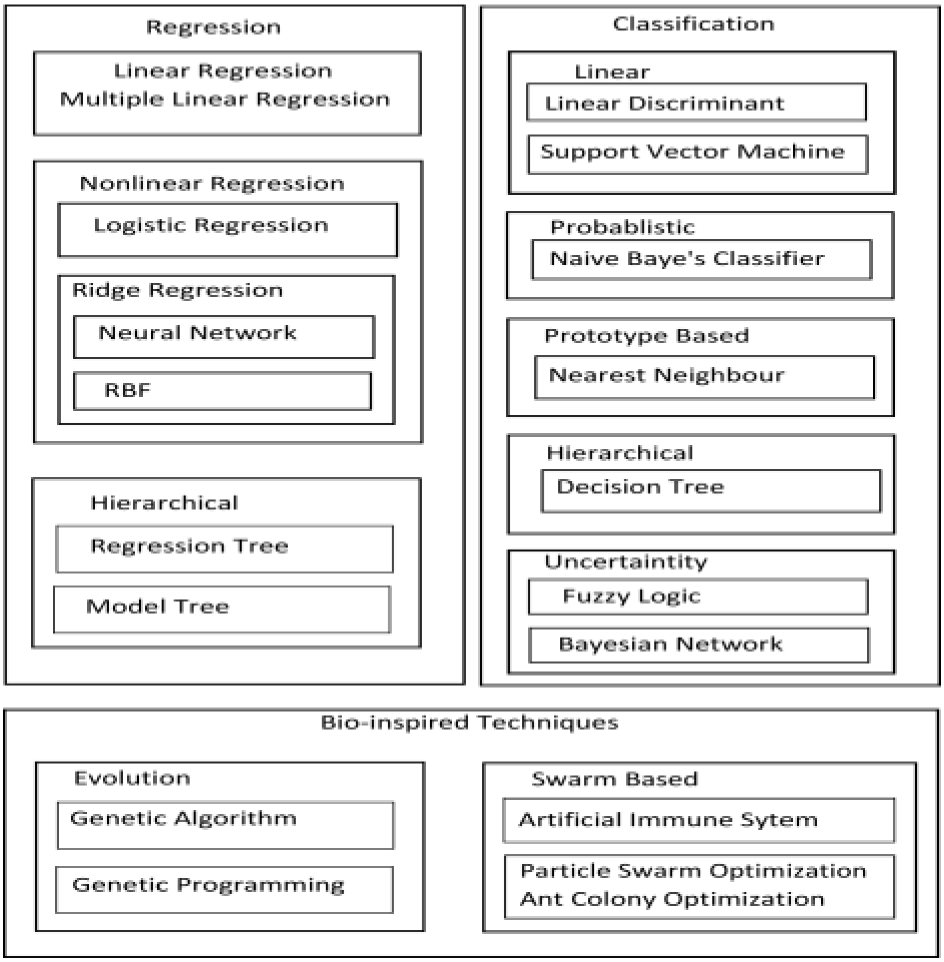

Figure 3:Taxonomy of Data Analysis Techniques

3.1 Linear and Logistic Regression

A statistical method for regression analysis is widely reported technique to construct fault prediction models.

In multiple linear regression (MLR) technique, relationship between two or more independent variables (x_1, x_2 ...x_k) with a dependent variable (y) is determined. The developed model can be viewed as Data = Fit + Residual.

To fit the model (i.e. to find regression coefficients) the ordinary least square method (OLS) is performed minimizing the squared distance between predicted and actual values, and the value of the relationship computed by the model can be predicted from residuals (Uysal and Guvenir, 1999; Yan and Su, 2009).

Logistic regression works like linear regression, except for the fact that independent variables may be categorical, and the response is a dichotomous outcome ranging from 0 to 1(Runkler, 2012).

Alshayeb and Li (Alshayeb and Li, 2003) established the relationship between OO metrics selected from Chidamber and Kemerer (CK) metrics suite (Chidamber and Kemerer, 1994) and development/maintenance efforts like Lines Changed (LC), Lines Added (LA), and Lines Deleted (LD) using Multiple Linear Regression (MLR). Even so, such a relationship was limited to short-cycled agile process and was found ineffective in the long-cycled framework process..

Basili *et al.* basili(Basili et al., 1996) used logistic regression to analyse the relationship between OO metrics and fault-proneness of classes during the early phases of the life-cycle. They had evaluated each metric in isolation using the univariate method, augmented by multi-variate regression to evaluate the predictive capability of those metrics. The outcomes of this study were validated with the data gathered from eight medium-sized software modules developed in C++.

Briand *et al.* (Briand et al., 2000) used logistic regression to the subset of OO metrics in the development of fault prediction model, owing to the fact that many OO metrics capture similar dimensions of measurement. The investigations were made with 28 coupling measures, ten cohesion measures, and 11 inheritance measures. Their work concluded the prevalence of coupling and inheritance measures and cohesion measures were found ineffectual.

Emam *et al.* (El Emam et al., 2001) illustrated the impact of confounding effect of class size in validation studies using logistic regression. Their study considered CK metrics and a subset of the Lorenz and Kidd metrics (Lorenz and Kidd, 1994) for a large C++ telecommunications framework. Supporting their argument, they suggested an Export Coupling (EC) metric and statistically established its strong association with fault-proneness.

Marcus *et al.* (Marcus et al., 2008) employed logistic regression accompanied by principal component analysis (PCA) on three open source software systems in support of their new measure for class cohesion: Conceptual Cohesion of Classes (C3). Their work concluded the superiority of the C3 metric over existing structural metrics.

3.2 SVM and Instance-based Learning

A Support Vector Machine (SVM) optimally separates data points into two categories using a kernel function. Model thus developed using an appropriate kernel function is closely related to the neural network and generalize well, though starting with a small training sample. This engenders SVM a suitable technique to develop fault prediction models, where the information of complexity metrics in the early phase of SDLC is very limited.

Elish *et al.* el(Elish and Elish, 2008) measured performance of Support vector machine (SVM) in classifying faults-prone software modules employing four publicly available NASA data sets. These data sets were derived from software projects developed in the different programming languages (C, C++, and Java). The predictive accuracy of the models developed through SVM with 21 static module level metrics with 10 fold cross validation was compared against eight other statistical and machine learning techniques (LR, KNN, RBF, MLP, NB, BBN, RF, and DT)[1] SVM showed superior performance of recall measures whilst also maintaining significant high values of F-measures.

Xing *et al.* (Xing et al., 2005) explored the utilization of SVM and its extended form (transductive SVM i.e. TSVM) on a random sample of 390(40000 lines of code) routines of a medical imaging software developed in Pascal, FORTRAN, assembly, and PL/M. A total of eleven complexity metrics was considered for model development. When compared to Quadratic discriminant analysis (QDA) as a classifier blended with PCA as the feature selection technique, SVM with RBF as the kernel trick based classifiers were reported to result in improved classification accuracy measures.

Di Martino *et al.* d(Di Martino et al., 2011) confirmed the advantages of using SVM as the linear classifier. However, they reasoned the applicability of SVM for non-linear classification. The parameters of underlying kernel function ought to be tuned by using the statistics of dataset. For example, in case of RBF as the kernel function,

[1] LR : Logistic regression, KNN: K-nearest neighbour, RBF: Radial basis function, MLP: Multi-layer perceptron, BBN: Bayesian belief network, NB: Naive Bayes, RF: Random forest, DT: Decision tree.

parameters like C (penalty factor for misclassified points) and γ (radius of the RBF) have an impact on classification accuracy. They recommended a genetic algorithm (GA) based approach to tune these parameters optimally for the dataset of Jedit software module available in the PROMISE data repository. The conclusion derived make evident the higher performance of the SVM models combined with GA.

3.3 Bayesian and Evidence-based Statistics

A Bayesian network (BN) represents an acyclic graph that embodies the joint probability distribution of a set of random variables. It models the casual influences on the problem and has not been explored in depth in the software measurement field, particularly in the predictive analytics of fault prediction model development. Construction of BN requires the modeling of qualitative influences in a domain through graphs and after that assignment of probabilities to each node in the representation.

Pai *et al.* (Pai and Dugan, 2007) developed BN by taking all products, process and another source of information accounting for fault introduction in software into consideration. Mining of product and process metrics data generates an individual BN structure. These different BN structures estimate external quality metrics like Fault content, Fault Proneness, reliability, etc. to predict the overall quality of software. This study summarizes contradictory, but interesting results. Significance of WMC, CBO, RFC, and SLOC metrics, with MLR as the mechanism to construct BN, supports the results reported by Gyimothy et. al. (Gyimothy et al., 2005) and insignificant metrics include DIT and NOC metrics.

Fenton *et al.* (Fenton et al., 2002) developed a toolkit AgeneRisk (available at http://www.agenarisk.com) to generate a dynamic Bayesian network that allows the construction of causal models to any phase of software Life cycle. The utilization of toolkit exhibited significantly improved and validated predictive accuracy in a trail of 30 different projects.

Bai *et al.* (Bai et al., 2005) developed a Markov Bayesian Network (MBN) to incorporate dynamic change in the model parameters of BN. To develop MBN, core ingredients shown are; initial distribution of defects computed from the data set, distribution of failure time and distribution of the number of defects removed over time. Their results concluded enhanced performance compared to traditional JelinskiMoranda model (JM model) and GoelOkumoto NHPP model (GO model).

Dejaeger *et al.* (Dejaeger et al., 2012) studied 15 different Bayesian Network (BN) classifiers using

NASA and Eclipse foundation data set and inferred that a general Bayesian network can be outperformed by the naive Bayes classifier when expanded with different augmentation operators like Tree augmenter, Forest augmenter, and selectively augment with and without discarding.

3.4 Additive Models and Trees

A classification and regression tree (CART) is a treelike representation of a succession of decisions involved. Each internal node encapsulates a decision taken to carry out subsequent predictions(Death and Fabricius, 2000; Dvzeroski and Drumm, 2003). In a classification tree (decision tree), labels are associated with the leaves, whereas, in the regression tree, the actual numerical value of the response variable is assigned to the leaf (Breiman et al., 1993). Model trees are an extension of regression trees that unite a linear model with each of the leaves instead of merely a numerical value (Frank et al., 1998; Quinlan, 1992).

The regression tree model for fault prediction was first reported by Gokhale and Lyu (Gokhale and Lyu, 1997). Since then a large number of studies have used these trees-based regression techniques, relevant amongst them are following:

Khoshgoftaar *et al.* kho(Khoshgoftaar et al., 2002) illustrated the effectiveness of a regression tree algorithm to identify fault-prone modules for 4 consecutive releases of a large telecommunications system using 24 product and four execution metrics.

Bibi *et al.* (Bibi et al., 2008) performed regression via classification (RvC) by discretizing target variables to train the classification model, and then reversed the process to change the output, back into a numerical prediction.

In this study, they experimented with different classification algorithms viz IBk JRip, PART, J48, and SMO available in Weka environment (Witten and Frank, 2005) using Pekka data set of a commercial bank (Maxwell, 2002) to validate the superiority of RvC approach.

Guo *et al.* guo(Guo et al., 2004) statistically analysed the relative performance of random forest over logistic regression and discriminant analysis using five case studies on a NASA data set. Random forests are variations of the decision trees and in this study, they generate a large number of such trees with the training data to establish the preponderance of random forest empirically.

Chowdhury *et al.* (Chowdhury and Zulkernine, 2011) analysed techniques like C4.5 Decision Tree, random forests, and logistic regression. They used fifty-two releases of Mozilla Firefox, developed over a period of four years to compare predictive performances. Their study concluded that the majority of the vulnerability-prone files in Mozilla Firefox can be identified with these techniques well within the tolerable false positive rates.

3.5 Perceptron based Models

Neural networks are universal approximation category of nonlinear regression method based on the action of biological neurons. In general, the term "Neural Network" (NN) and "Artificial Neural Network" (ANN) belongs to a Multilayer Perceptron Network. Additional prototypes of neural network include Probabilistic Neural Networks (PNN), General Regression Neural Networks (GRNN), Ward neural network (WNN), Radial Basis Function (RBF), Recurrent Networks and Hybrid Networks etc (Yuhas and Ansari, 2012)[**Error! Reference source not found.**].

Zheng *et al.* (Zheng, 2010) took the severity of type II error into consideration to develop neural network-based predictive models. Type II error deals with the misclassification of defect-prone modules, whereas Type I error relates the misclassification of not-defect-prone ones. Neural Network with cost-sensitive Adaboost (boosting technique) (Runkler, 2012) manifested reduced number of such type II errors.

Khoshgoftaar *et al.* (Khoshgoftaar et al., 1997) first illustrated the utilisation of neural-network for EMERALD (Enhanced measurement for early risk assessment of latent defects), a joint project of Nortel and Bell Canada to improve the reliability of software. Their results manifested that neural manages Type II classification error efficiently compared to discriminant analyses.

Kanmani *et al.* (Kanmani et al., 2007) compared and analysed the performance of Back Propagation Neural Network (BPN) and Probabilistic Neural Network (PNN) to predict the fault-proneness of the C++ modules with conventional logistic regression using the data set generated from the software modules developed by the graduate students. This study empirically verified the robustness of the predictive accuracy of PNN using five quality parameters.

Thwin *et al.* (Thwin and Quah, 2003) analysed the comparative performance of ward neural network (WNN) and General Regression Neural network (GRNN) to predict count of defects in a class and the number of lines change per class. A WNN is a back propagation network with three slabs in the hidden layer having different activation functions. GRNN is one-pass learning and memory based network structure. This study reasoned the superior predictive ability of GRNN over compared to WNN.

3.6 Fuzzy Logic based Approaches

Fuzzy based models change the subjective knowledge into mathematically explorable terms and rules to create systems with a level of uncertainty.

The use of fuzzy logic in the modeling of various perspectives of software development process is increasingly achieving attention of researchers. Following is the concise summary of related contributions published in the literature;

So *et al.* (So et al., 2002) empirically analyzed the performance of fuzzy logic to predict fault-prone modules using inspection data. They built up an automated and scalable system that performs well, even if huge inspection data is not usable.

Pandey *et al.* (Pandey and Goyal, 2009) explored the effectiveness of fuzzy expert system in the prediction of the occurrence of faults after each phase of the software development life cycle (SDLC). Fuzzy inference system of their model employs eight reliability metrics collected for different phases of SDLC.

Xu *et al.* (Xu et al., 2008) demonstrated the inference ability of fuzzy expert system with limited facts available. Their study resulted in the maturation of a risk assessment framework following NASA standards.

Yang *et al.* (Yang et al., 2007) proposed a hybrid model of Neural and Fuzzy logic. This plan uses the knowledge derived from previous similar projects for training and efficiently deals with the data that is objective in nature.

Muzaffar *et al.* (Muzaffar and Ahmed, 2010) analysed the impact of de-fuzzification and membership functions in the conception of a fuzzy logic based system for software development effort.

Verma *et al.* (Verma and Sharma, 2010) proposed a fuzzy logic-based framework for development effort evaluation and reported increased performance on an artificial and live project data both. Their conclusions statistically establish the efficacy of fuzzy logic based system to manage the imprecision in the input data.

Aljahdali *et al.* (Aljahdali and Sheta, 2011) reported encouraging outcomes using fuzzy nonlinear regression in modelling accumulated faults in software modules.

3.7 Bio-inspired Techniques

Evolutionary techniques are bio-inspired meta-heuristic approaches and exhibit common characteristics (Back et al., 1997).

1. Execution of these techniques begins with a population of the candidate solution set constituting the search space.
2. A selection process identifies better solution through a derived fitness criteria depending upon the problem formulation.
3. New solutions evolve through mutation and recombination.

Azar *et al.* (Azar and Vybihal, 2011) optimized existing software quality estimation models using ant colony optimization (ACO) technique. ACO adapted with previously developed predictive models put to use a common domain and context-specific data for model construction. This permits to infer predictive models built for one dataset for new data. The result of this study concluded with the enhanced performance of ACO compared to C4.5 and random guessing techniques.

Khoshgoftaar *et al.* (Khoshgoftaar and Seliya, 2003) investigated the influence of genetic programming (GP) in developing decision trees to solve software quality classification problem whilst minimizing the cost of misclassification and the size of tree simultaneously. Two initial releases of large windows based embedded systems comprising of more than 27 million lines of codes generated dataset used in this study. The results concluded that GP based decision tree modelling accounts for greater flexibility in building optimal classification models.

Vandecruys *et al.* Vandecruys (Vandecruys et al., 2008) empirically verified the advantage of AntMiner+ classification process over C4.5, logistic regression and support vector machines using NASA data repository to predict faults in the software module. AntMiner+ is a classification method based on ACO and deduces a rule-based classification models from a dataset. The Implementation of AntMiner+ is accessible on the web (Refer http://www.antminerplus.com).

Bouktif *et al.* (Bouktif et al., 2010) trained predictive model parameters from already built models. In the proposed mechanism, new models develop through the genetic algorithm based combination and adaptation of the expertise already available in existing prediction models. The application of this mechanism with decision trees over NASA data achieved significantly improved selection of models.

Chiu *et al.* (Chiu, 2011) in one way extends the previous work of Bouktif et. al. [**Error! Reference source not found.**] and suggested an integrated decision network (IDN) wherein particle swarm optimisation (PSO) implements the combination and adaptation phases of the model development. In comparison to GA, PSO approach needs fewer complex operators, hence makes it more appropriate to design IDN. The derived results establish that the proposed mechanism outperforms individual software quality classification models and provides a deeper insight to decision makers.

Nature inspired computational techniques like the Artificial Immune system have been used in fault prediction and performance and are reportedly better than J48 classifiers (Catal and Diri, 2007). Search-based software engineering (SBSE), which utilizes nature-inspired techniques in empirical software engineering is an emerging field.

SBSE is gaining momentum with the advent of enhanced heuristic algorithms (Gay, 2010; Harman, 2010; Harman et al., 2012, 2009; Meziane and Vadera, 2010).

Studies indicated below points to the investigations, which take advantage of the combination of some of the techniques above and address other relevant aspects of software measurement:

Bibi *et al.* (Bibi et al., 2008) used a combination of classification and regression techniques by executing regression, via classification. Gyimothy *et al.* (Gyimothy et al., 2005) validated metrics for fault-proneness predictions in the "Bugzilla" database using a combination of regression and machine learning methods.

Nagappan *et al.* (Nagappan et al., 2006) provided an excellent step by step guide to develop quality predictors.

Beecham *et al.* (Beecham et al., 2008, 2006) and Kitchenham et.al. (Kitchenham et al., 2009, 2002) provide with notable systematic literature reviews (SLR) in empirical software engineering, along with an unfolded mechanism to administer a new, although other suitable literature reviews are also accessible (Biolchini et al., 2005; Petersen et al., 2008).

Menzies and Shepperd (Menzies and Shepperd, 2012) express their opinions about the sample size, applied statistical techniques and the conclusion stability of the published results in the editorial of the "*Special issue on repeatable results in software engineering prediction*". This premium editorial give emphasis on the reproducibility of the published results and infers the studies made by Dybaa *et al.* (Dybaa et al., 2006) and Easterbrook *et al.* (Easterbrook et al., 2008). Further, Singer *et al.* (Singer and Vinson, 2002) recognizes ethical and legal issues implicated in empirical software engineering.

4. Fault Prediction Using Metrics Combination

The Software Development Life Cycle transforms artifacts like a software requirement specification (SRS) to a final product. The nature of the relationship between artifacts and suitable transformation leads to a large number of the resultant artifacts (Raffo et al., 2000). Combination of metrics, selected from different phases of the software development lifecycle, results in improved accuracy of predictive models.

However, while combining several metrics; the issue of multi–collinearity arises due to inter-correlation among the metrics. To overcome this, various feature selection techniques like Principal Components Analysis (PCA) may be used. With PCA, a smaller number of uncorrelated linear combinations of metrics can be obtained na(Nagappan et al., 2006).

Following are the notable works in this field, although somewhat limited in number:

1. Wahyudin *et al.* (Wahyudin et al., 2008) examined the

combined effects of product and project metrics in the development of an improved predictive model. Their study used project metrics collected from Apache MyFaces project family over a span of two years. Through, correlation analysis, selected project metrics revealed a strong correlation between product metrics. To reduce the dimensionality of the combination of metrics, stepwise regression was applied. Their work shows the importance of the combination of metrics, without deliberating interaction between metrics.

2. D'Ambros *et al.* (DAmbros et al., 2012) Ambros statistically analyzed the benefits of utilizing a combination of source code metrics and other metrics derived using information theory to predict bugs. The same authors earlier showed the comparative advantages of using the combination of CK and other object-oriented metrics (DAmbros et al., 2010). They created a bug prediction data set and made it public. The same data set is being used in our research.
3. Lee *et al.* (Lee et al., 2011) proposed 56 micro interaction metrics (MIMs) capturing developer's behavioral pattern stored in Mylyn data. Metrics associated with behavioral pattern measures developer interaction with the development environment, for example, file editing, time spent on an event, etc. they build both classification and regression models using MIM in isolation and as well as in combination with other traditional metrics and empirically analyzed their effect on software quality. This experimental data of their study is freely available for future research purposes.

This combined metrics approach of fault prediction may utilize different metrics selected from within a single project or across multiple projects. Most metrics developed for process, products and people relate to one another; therefore their combination will lead to the issue of appropriate selection of candidate metrics and take their interaction effect into account.

5. Conclusion

This paper delineates metrics, metrics suite and their usage to the applied data analytic techniques. Although developments of models make use of different kinds of metrics, the review of the literature presented here essentially focuses on the Object oriented metrics. In comparison to, procedural language based system, Object Oriented (OO) technology based systems introduce new abstractions and building blocks. Therefore, development of the new set of metrics and fault prediction models will foster quality in the developed software. The advantages of combining metrics, while implementing a metrics program in an organisation needs further investigation.

References

Aggarwal, K.K., Singh, Y., Kaur, A., Malhotra, R., 2006. Empirical Study of Object-Oriented Metrics. Journal of Object Technology 5, 149–173.

Aggarwal, K.K., Singh, Y., Kaur, A., Malhotra, R., 2009. Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated case study. Software Process: Improvement and Practice 14, 39–62.

Aljahdali, S., Sheta, A.F., 2011. Predicting the Reliability of Software Systems Using Fuzzy Logic, in: Information Technology: New Generations (ITNG), 2011 Eighth International Conference on. pp. 36–40.

Alshayeb, M., Li, W., 2003. An empirical validation of object-oriented metrics in two different iterative software processes. Software Engineering, IEEE Transactions on 29, 1043–1049.

Anh, N.D., 2010. The impact of design complexity on software cost and quality.

Arisholm, E., Briand, L.C., Johannessen, E.B., 2010. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. Journal of Systems and Software 83, 2–17.

Azar, D., Vybihal, J., 2011. An ant colony optimization algorithm to improve software quality prediction models: Case of class stability. Information and Software Technology 53, 388–393.

Babic, D., 2012. Adaptive Software Fault Prediction Approach Using Object-Oriented Metrics.

Back, T., Fogel, D.B., Michalewicz, Z., 1997. Handbook of evolutionary computation. IOP Publishing Ltd.

Bai, C.G., Hu, Q.P., Xie, M., Ng, S.H., 2005. Software failure prediction based on a Markov Bayesian network model. Journal of Systems and Software 74, 275–282.

Bansiya, J., Davis, C.G., 2002. A hierarchical model for object-oriented design quality assessment. Software Engineering, IEEE Transactions on 28, 4–17.

Basili, V.R., Briand, L.C., Melo, W.L., 1996. A validation of object-oriented design metrics as quality indicators. Software Engineering, IEEE Transactions on 22, 751–761.

Beecham, S., Baddoo, N., Hall, T., Robinson, H., Sharp, H., 2006. Protocol for a systematic literature review of motivation in software engineering.

Beecham, S., Baddoo, N., Hall, T., Robinson, H., Sharp, H., 2008. Motivation in Software Engineering: A systematic literature review. Information and Software Technology 50, 860–878.

Bibi, S., Tsoumakas, G., Stamelos, I., Vlahavas, I., 2008. Regression via Classification applied on software defect estimation. Expert Systems with Applications 34, 2091–2101.

Biolchini, J., Mian, P.G., Natali, A.C.C., Travassos, G.H., 2005. Systematic review in software engineering. System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES 679.

Bouktif, S., Ahmed, F., Khalil, I., Antoniol, G., 2010. A novel composite model approach to improve software quality prediction. Information and Software Technology 52, 1298–1311.

Brady, A., Menzies, T., 2010. Case-based reasoning vs parametric models for software quality optimization, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 3.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1993. Classification and Regression Trees, Wadsworth International Group, Belmont, CA, 1984. There is no corresponding record for this reference 1–359.

Briand, L.C., Daly, J., Porter, V., Wust, J., 1998. A comprehensive empirical validation of design measures for object-oriented systems, in: Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International. pp. 246–257.

Briand, L.C., Wust, J., Daly, J.W., Porter, V., 2000. Exploring the relationships between design measures and software quality in object-oriented systems. Journal of Systems and Software 51, 245–273.

Caglayan, B., Tosun, A., Miranskyy, A., Bener, A., Ruffolo, N., 2010. Usage of multiple prediction models based on defect categories, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 8.

Carapucca, R., Others, 1994. Candidate metrics for object-oriented software within a taxonomy framework. Journal of Systems and Software 26, 87–96.

Catal, C., 2011. Software fault prediction: A literature review and current trends. Expert Systems with Applications 38, 4626–4636.

Catal, C., Diri, B., 2007. Software fault prediction with object-oriented metrics based artificial immune recognition system.

Product-Focused Software Process Improvement 300–314.

Catal, C., Diri, B., 2009. A systematic review of software fault prediction studies. Expert Systems with Applications 36, 7346–7354.

Catal, C., Sevim, U., Diri, B., 2011. Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm. Expert Systems with Applications 38, 2347–2353.

Chidamber, S.R., Kemerer, C.F., 1994. A metrics suite for object oriented design. Software Engineering, IEEE Transactions on 20, 476–493.

Chiu, N., 2011. Combining techniques for software quality classification: An integrated decision network approach. Expert Systems with Applications 38, 4618–4625.

Chowdhury, I., Zulkernine, M., 2011. Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. Journal of Systems Architecture 57, 294–313.

Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E., 2010. How effective is tabu search to configure support vector regression for effort estimation?, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 4.

Couto, C., Montandon, J.E., Silva, C., Valente, M.T., 2012. Static correspondence and correlation between field defects and warnings reported by a bug finding tool. Software Quality Journal 1–17.

DAmbros, M., Lanza, M., Robbes, R., 2010. An extensive comparison of bug prediction approaches, in: Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on. pp. 31–41.

DAmbros, M., Lanza, M., Robbes, R., 2012. Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empirical Software Engineering 17, 531–577.

Death, G., Fabricius, K.E., 2000. Classification and regression trees: a powerful yet simple technique for ecological data analysis. Ecology 81, 3178–3192.

Dejaeger, K., Verbraken, T., Baesens, B., 2012. Towards comprehensible software fault prediction models using Bayesian network classifiers. Software Engineering, IEEE Transactions on.

Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., 2011. A genetic algorithm to configure support vector machines for predicting fault-prone components, in: Product-Focused Software Process Improvement. Springer, pp. 247–261.

Dick, S., Meeks, A., Last, M., Bunke, H., Kandel, A., 2004. Data mining in software metrics databases. Fuzzy Sets and Systems 145, 81–110.

Dvzeroski, S., Drumm, D., 2003. Using regression trees to identify the habitat preference of the sea cucumber (< i> Holothuria leucospilota</i>) on Rarotonga, Cook Islands. Ecological Modelling 170, 219–226.

Dybaa, T., Kampenes, V.B., Sjo berg, D.I.K., 2006. A systematic review of statistical power in software engineering experiments. Information and Software Technology 48, 745–755.

E Abreu, F., Melo, W., 1996. Evaluating the impact of object-oriented design on software quality, in: Software Metrics Symposium, 1996., Proceedings of the 3rd International. pp. 90–99.

Easterbrook, S., Singer, J., Storey, M.-A., Damian, D., 2008. Selecting empirical methods for software engineering research, in: Guide to Advanced Empirical Software Engineering. Springer, pp. 285–311.

Ebert, C., Dumke, R., 2007. Measurement Foundations. Software Measurement: Establish Extract Evaluate and Execute 41–72.

El Emam, K., Benlarbi, S., Goel, N., Rai, S.N., 2001. The confounding effect of class size on the validity of object-oriented metrics. Software Engineering, IEEE Transactions on 27, 630–650.

Elish, K.O., Elish, M.O., 2008. Predicting defect-prone software modules using support vector machines. Journal of Systems and Software 81, 649–660.

Fenton, N., Krause, P., Neil, M., 2002. Software measurement: Uncertainty and causal modeling. Software, IEEE 19, 116–122.

Frank, E., Wang, Y., Inglis, S., Holmes, G., Witten, I.H., 1998. Using model trees for classification. Machine Learning 32, 63–76.

Gay, G., 2010. A baseline method for search-based software engineering, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 2.

Gokhale, S.S., Lyu, M.R., 1997. Regression tree modeling for the prediction of software quality, in: Proceedings of the Third ISSAT International Conference on Reliability and Quality in Design. pp. 31–36.

Guo, L., Ma, Y., Cukic, B., Singh, H., 2004. Robust prediction of fault-proneness by random forests, in: Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on. pp. 417–428.

Gyimothy, T., Ferenc, R., Siket, I., 2005. Empirical validation of object-oriented metrics on open source software for fault prediction. Software Engineering, IEEE Transactions on 31, 897–910.

Hall, T., Beecham, S., Bowes, D., Gray, D., Counsell, S., 2011. A systematic review of fault prediction performance in software engineering. Software Engineering, IEEE Transactions on.

Harman, M., 2010. The relationship between search based software engineering and predictive modeling, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 1.

Harman, M., Mansouri, S.A., Zhang, Y., 2009. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Department of Computer Science, King's College London, Tech. Rep. TR-09-03.

Harman, M., McMinn, P., de Souza, J.T., Yoo, S., 2012. Search based software engineering: Techniques, taxonomy, tutorial, in: Empirical Software Engineering and Verification. Springer, pp. 1–59.

Harrison, R., Counsell, S.J., Nithi, R.V., 1998. An evaluation of the MOOD set of object-oriented software metrics. Software Engineering, IEEE Transactions on 24, 491–496.

Hitz, M., Montazeri, B., 1995. Measuring coupling and cohesion in object-oriented systems, in: Proceedings of the International Symposium on Applied Corporate Computing. pp. 75–76.

Hong, Y., Kim, W., Joo, J., 2010. Prediction of defect distribution based on project characteristics for proactive project management, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 15.

Janes, A., Scotto, M., Pedrycz, W., Russo, B., Stefanovic, M., Succi, G., 2006. Identification of defect-prone classes in telecommunication software systems using design metrics. Information Sciences 176, 3711–3734.

Jones, T.C., 2008. Applied Software Measurement: Global Analysis of Productivity and Quality, 3E.

Kaner, C., Bond, W.P., 2004. Software engineering metrics: What do they measure and how do we know? methodology 8, 6.

Kanmani, S., Uthariaraj, V.R., Sankaranarayanan, V., Thambidurai, P., 2007. Object-oriented software fault prediction using neural networks. Information and Software Technology 49, 483–492.

Khoshgoftaar, T.M., Allen, E.B., Deng, J., 2002. Using regression trees to classify fault-prone software modules. Reliability, IEEE Transactions on 51, 455–462.

Khoshgoftaar, T.M., Allen, E.B., Hudepohl, J.P., Aud, S.J., 1997. Application of neural networks to software quality modeling of a very large telecommunications system. Neural Networks, IEEE Transactions on 8, 902–909.

Khoshgoftaar, T.M., Seliya, N., 2003. Fault prediction modeling for software quality estimation: Comparing commonly used techniques. Empirical Software Engineering 8, 255–283.

Khoshgoftaar, T.M., Seliya, N., Sundaresh, N., 2006. An empirical study of predicting software faults with case-based reasoning. Software Quality Journal 14, 85–111.

Kitchenham, B., Brereton, P., Budgen, D., Turner, M., Bailey, J., Linkman, S., 2009. Systematic literature reviews in software engineering-a systematic literature review. Information and software technology 51, 7–15.

Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J., 2002. Preliminary guidelines for empirical research in software engineering. Software Engineering, IEEE Transactions on 28, 721–734.

Koru, A.G., Liu, H., 2005. Building effective defect-prediction models in practice. Software, IEEE 22, 23–29.

Laird, L.M., Brennan, M.C., 2006. Software measurement and estimation: a practical approach. John Wiley and Sons.

Lanubile, F., Visaggio, G., 1997. Evaluating predictive quality models derived from software measures: lessons learned. Journal of Systems and Software 38, 225–234.

Lavazza, L., Robiolo, G., 2010. The role of the measure of functional complexity in effort estimation, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 6.

Lee, T., Nam, J., Han, D., Kim, S., In, H.P., 2011. Micro interaction metrics for defect prediction., in: SIGSOFT FSE. pp. 311–321.

Lessmann, S., Baesens, B., Mues, C., Pietsch, S., 2008. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. Software Engineering, IEEE Transactions on 34, 485–496.

Li, W., Henry, S., 1993. Object-oriented metrics that predict maintainability. Journal of systems and software 23, 111–122.

Li, W., Henry, S., Selig, C., 1991. Measuring Ada Design to Predict Maintainability, in: 9th Annual National Conference on Ada Technology. pp. 107–113.

Lorenz, M., Kidd, J., 1994. Object-oriented software metrics: a practical guide. Prentice-Hall, Inc.

Luo, Y., Ben, K., Mi, L., 2010. Software metrics reduction for fault-proneness prediction of software modules, in: Network and Parallel Computing. Springer, pp. 432–441.

Mair, C., Shepperd, M., 2011. Human judgement and software metrics: vision for the future, in: Proceedings of the 2nd International Workshop on Emerging Trends in Software Metrics. pp. 81–84.

Malhotra, R., Kaur, A., Singh, Y., 2010. Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines. International Journal of System Assurance Engineering and Management 1, 269–281.

Marcus, A., Poshyvanyk, D., Ferenc, R., 2008. Using the conceptual cohesion of classes for fault prediction in object-oriented systems. Software Engineering, IEEE Transactions on 34, 287–300.

Matsumoto, S., Kamei, Y., Monden, A., Matsumoto, K., Nakamura, M., 2010. An analysis of developer metrics for fault prediction, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 18.

Maxwell, K., 2002. Applied statistics for software managers. Prentice Hall.

McCabe, T.J., 1976. A complexity measure. Software Engineering, IEEE Transactions on 1, 308–320.

Mende, T., 2010. Replication of defect prediction studies: problems, pitfalls and recommendations, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering. p. 5.

Menzies, T., Milton, Z., Turhan, B., Cukic, B., Jiang, Y., Bener, A., 2010. Defect prediction from static code features: current results, limitations, new approaches. Automated Software Engineering 17, 375–407.

Menzies, T., Shepperd, M., 2012. Special issue on repeatable results in software engineering prediction. Empirical Software Engineering 17, 1–17.

Mertik, M., Lenic, M., Stiglic, G., Kokol, P., 2006. Estimating software quality with advanced data mining techniques, in: Software Engineering Advances, International Conference on. p. 19.

Meziane, F., Vadera, S., 2010. Artificial intelligence applications for improved software engineering development: new prospects. Information Science Reference.

Michura, J., Capretz, M.A.M., Wang, S., 2013. Extension of Object-Oriented Metrics Suite for Software Maintenance. ISRN Software Engineering 2013.

Misra, S., 2011. Evaluation Criteria for Object-oriented Metrics. Acta Polytechnica Hungarica 8, 110–136.

Misra, S., Adewumi, A., 2014. Object-Oriented Cognitive Complexity Measures: An Analysis. Handbook of Research on Innovations in Systems and Software Engineering 150.

Misra, S., Koyuncu, M., Crasso, M., Mateos, C., Zunino, A., 2012. A suite of cognitive complexity metrics, in: Computational Science and Its Applications-ICCSA 2012. Springer, pp. 234–247.

Muzaffar, Z., Ahmed, M.A., 2010. Software development effort prediction: A study on the factors impacting the accuracy of fuzzy logic systems. Information and Software Technology 52, 92–109.

Nagappan, N., Ball, T., Zeller, A., 2006. Mining metrics to predict component failures, in: Proceedings of the 28th International Conference on Software Engineering. pp. 452–461.

Pai, G.J., Dugan, J.B., 2007. Empirical analysis of software fault content and fault proneness using Bayesian methods. Software Engineering, IEEE Transactions on 33, 675–686.

Pandey, A.K., Goyal, N.K., 2009. A Fuzzy Model for Early Software Fault Prediction Using Process Maturity and Software Metrics. International Journal of Electronics Engineering 1, 239–245.

Perry, D.E., Porter, A.A., Votta, L.G., 2000. Empirical studies of software engineering: a roadmap, in: Proceedings of the Conference on The Future of Software Engineering. pp. 345–355.

Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering. p. 1.

Quinlan, J.R., 1992. Learning with continuous classes, in: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence. pp. 343–348.

Radjenovic, D., Herico, M., Torkar, R., Zivkovic, A., 2013. Software fault prediction metrics: A systematic literature review. Information and Software Technology 55, 1397–1418.

Raffo, D., Harrison, W., Vandeville, J., 2000. Coordinating models and metrics to manage software projects. Software Process: Improvement and Practice 5, 159–168.

Raj Kiran, N., Ravi, V., 2008. Software reliability prediction by soft computing techniques. Journal of Systems and Software 81, 576–583.

Rodriguez, D., Herraiz, I., Harrison, R., 2012. On software engineering repositories and their open problems, in: Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2012 First International Workshop on. pp. 52–56.

Runeson, P., Andersson, C., Thelin, T., Andrews, A., Berling, T., 2006. What do we know about defect detection methods?[software testing]. Software, IEEE 23, 82–90.

Runkler, T.A., 2012. Data Analytics: Models and Algorithms for Intelligent Data Analysis. Vieweg Teubner Verlag.

Seaman, C.B., 1999. Qualitative methods in empirical studies of software engineering. Software Engineering, IEEE Transactions on 25, 557–572.

Shepard, T., Lamb, M., Kelly, D., 2001. More testing should be taught. Communications of the ACM 44, 103–108.

Singer, J., Vinson, N., 2002. Ethical issues in empirical studies of software engineering.

So, S.S., Cha, S.D., Kwon, Y.R., 2002. Empirical evaluation of a fuzzy logic-based software quality prediction model. Fuzzy Sets and Systems 127, 199–208.

Succi, G., Pedrycz, W., Stefanovic, M., Miller, J., 2003. Practical assessment of the models for identification of defect-prone classes in object-oriented commercial systems using design metrics. Journal of Systems and Software 65, 1–12.

Tang, M.-H., Kao, M.-H., Chen, M.-H., 1999. An empirical study on object-oriented metrics, in: Software Metrics Symposium, 1999. Proceedings. Sixth International. pp. 242–249.

Tegarden, D.P., Sheetz, S.D., Monarchi, D.E., 1995. A software complexity model of object-oriented systems. Decision Support Systems 13, 241–262.

Thwin, M.M.T., Quah, T.-S., 2003. Application of neural networks for software quality prediction using object-oriented metrics, in: Journal of Systems and Software. Elsevier, pp. 147–156.

Tichy, W.F., 1998. Should computer scientists experiment more? Computer 31, 32–40.

Uysal, I., Guvenir, H.A., 1999. An overview of regression techniques for knowledge discovery. Knowledge Engineering Review 14, 319–340.

Vandecruys, O., Martens, D., Baesens, B., Mues, C., De Backer, M., Haesen, R., 2008. Mining software repositories for comprehensible software fault prediction models. Journal of Systems and software 81, 823–839.

Verma, H.K., Sharma, V., 2010. Handling imprecision in inputs using fuzzy logic to predict effort in software development, in: Advance Computing Conference (IACC), 2010 IEEE 2nd International. pp. 436–442.

Wagner, S., 2013. Quality Planning, in: Software Product Quality Control. Springer, pp. 91–110.

Wahyudin, D., Schatten, A., Winkler, D., Tjoa, A.M., Biffl, S., 2008. Defect Prediction using Combined Product and Project Metrics-A Case Study from the Open Source, in: Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference. pp. 207–215.

Wang, Y., Shao, J., 2003. Measurement of the cognitive functional complexity of software, in: Cognitive Informatics, 2003. Proceedings. The Second IEEE International Conference on. pp. 67–74.

Witten, I.H., Frank, E., 2005. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

Xing, F., Guo, P., Lyu, M.R., 2005. A novel method for early software quality prediction based on support vector machine, in: Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on. p. 10–pp.

Xu, J., Ho, D., Capretz, L.F., 2008. An empirical validation of object-oriented design metrics for fault prediction. Journal of Computer Science 4, 571.

Yan, X., Su, X.G., 2009. Linear regression analysis: theory and computing. World Scientific Publishing Company.

Yang, B., Yao, L., Huang, H.-Z., 2007. Early software quality prediction based on a fuzzy neural network model, in: Natural Computation, 2007. ICNC 2007. Third International Conference on. pp. 760–764.

Yuhas, B., Ansari, N., 2012. Neural networks in telecommunications. Springer Publishing Company, Incorporated.

Zheng, J., 2010. Cost-sensitive boosting neural networks for software defect prediction. Expert Systems with Applications 37, 4537–4543.