Artif Intell Rev
DOI 10.1007/s10462-017-9563-5

A study on software fault prediction techniques

Santosh S. Rathore[1] · **Sandeep Kumar**[1]

© Springer Science+Business Media Dordrecht 2017

Abstract Software fault prediction aims to identify fault-prone software modules by using some underlying properties of the software project before the actual testing process begins. It helps in obtaining desired software quality with optimized cost and effort. Initially, this paper provides an overview of the software fault prediction process. Next, different dimensions of software fault prediction process are explored and discussed. This review aims to help with the understanding of various elements associated with fault prediction process and to explore various issues involved in the software fault prediction. We search through various digital libraries and identify all the relevant papers published since 1993. The review of these papers are grouped into three classes: software metrics, fault prediction techniques, and data quality issues. For each of the class, taxonomical classification of different techniques and our observations have also been presented. The review and summarization in the tabular form are also given. At the end of the paper, the statistical analysis, observations, challenges, and future directions of software fault prediction have been discussed.

Keywords Software fault prediction · Software metrics · Fault prediction techniques · Software fault datasets · Taxonomic classification

1 Introduction and motivation

Software quality assurance (SQA) consists of monitoring and controlling the software development process to ensure the desired software quality at a lower cost. It may include the application of formal code inspections, code walkthroughs, software testing, and software fault prediction (Adrion et al. 1982; Johnson Jr and Malek 1988). Software fault prediction

✉ Sandeep Kumar
sandeepkumargarg@gmail.com; sgargfec@iitr.ac.in

Santosh S. Rathore
santosh.srathore@gmail.com

[1] Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee, India

Published online: 30 May 2017

aims to facilitate the allocation of limited SQA resources optimally and economically by prior prediction of the fault-proneness of software modules (Menzies et al. 2010). The potential of software fault prediction to identify faulty software modules early in the development life cycle has gained considerable attention over the last two decades. Earlier fault prediction studies used a wide range of classification algorithms to predict the fault-proneness of software modules. The results of these studies showed the limited capability of the algorithm's potentials and thus questioning their dependability for software fault prediction (Catal 2011). The prediction accuracy of fault-prediction techniques found to be considerably lower, ranging between 70% and 85%, with high misclassification rate (Venkata et al. 2006; Elish and Elish 2008; Guo et al. 2003). An important concern related to software fault prediction is the lack of suitable performance evaluation measures that can assess the capability of fault prediction models (Jiang et al. 2008). Another concern is about the unequal distribution of faults in the software fault datasets that may lead to the biased learning (Menzies et al. 2010). Moreover, some issues such as the choice of software metrics to be included in fault prediction models, effect of context on prediction performance, cost-effectiveness of fault prediction models, and prediction of fault density, need further investigation. Recently, a number of software project dataset repositories have became publicly available such as NASA Metrics Data Program[1] and PROMISE Data Repository.[2] The availability of these repositories has encouraged undertaking more investigations and opening up new areas of applications. Therefore, the review of state-of-art in this area can be useful to the research community.

Among others, some of the literature reviews reported in this area are Hall et al. (2012), Radjenovic et al. (2013), Kitchenham (2010). In Kitchenham (2010) reported a mapping study of software metrics. The study mainly focused on identifying and categorizing influential software metric research between the periods of 2000 and 2005. Further, author assessed the possibility of aggregating the results of various research papers to draw the useful conclusions about the capability of software metrics. Author found that lot of studies have been performed to validate software metrics for software fault prediction. However, lack of empirical investigations and not use of proper data analysis techniques made it difficult to draw any generalized conclusion. The review study did not provide any assessment of other dimensions of the software fault prediction process. Only studies validating software metrics for software fault prediction have been presented. The review only focused on the studies between 2000 and 2005. However, after 2005 (since the availability of the PROMISE data repository), lots of research have been done over the open source projects, which is missing in this review study. In Catal (2011), investigated 90 papers related to software fault prediction techniques published during the period from 1990 to 2009 and grouped them on the basis of the year of publication. The study has investigated and evaluated various techniques for their potential to predict fault-prone software modules. His appraisal of earlier studies has included the analysis of software metrics and fault prediction techniques. Later, the current developments in fault prediction techniques have been introduced and discussed. However, the review study discussed and investigated the methods and techniques used to build the fault prediction model without considering any context or environment variable over which validation studies were performed.

Hall et al. (2012) have presented a review study on fault prediction performance in software engineering. The objective of the study was to appraise the context of fault prediction model, used software metrics, dependent variables, and fault prediction techniques on the

[1] NASA Data Repository, http://mdp.ivv.nasa.gov.

[2] PROMISE Data Repository, http://openscience.us/repo/.

performance of software fault prediction. The review included 36 studies published between 2000 and 2010. According to the study, fault prediction techniques such as Naive Bayes and Logistic Regression have produced better fault prediction results, while techniques such as SVM and C4.5 did not perform well. Similarly, for independent variables, it was found that object-oriented (OO) metrics produced better fault prediction results compared to other metrics such as LOC and complexity metrics. This work also presented the quantitative and qualitative models to assess the software metrics, context of fault prediction, and fault prediction techniques. However, they did not provide any details about how various factors of software fault prediction are interrelated to and different from each other. Moreover, no taxonomical classification of software fault prediction components has been provided. In another study, Radjenovic et al. (2013) presented a review study related to the analysis of software metrics for fault prediction. They found that object-oriented metrics (49%) were the highest used by researchers, followed by the traditional source code metrics (27%) and process metrics (24%) as the second and third highest used metrics. They concluded that it is more beneficial to use object-oriented and process metrics for fault prediction compare to traditional size or complexity metrics. Furthermore, they added that process metrics produced significantly better results in predicting post-release faults compared to static code metrics. Radjenovic et al. extended the Kitchenham's review work (Kitchenham 2010) and assessed the applicability of software metrics for fault prediction. However, they did not incorporate other aspects of the software fault prediction that may affect the applicability of software metrics.

Recently, Kamei and Shihab (2016) presented a study that provides a brief overview of software fault prediction and its components. The study highlighted accomplishments made in software fault prediction as well as discussed current trends in the area. Additionally, some of the future challenges for software fault prediction have been identified and discussed. However, the study did not provide details of various works on software fault prediction. Also, advantages and drawbacks of existing works on software fault prediction have not been provided.

In this paper, we explore various dimensions of software fault prediction process and analyze their influence on the prediction performance. The contribution of this paper can be summarized as follows. Various available review studies such as Catal (2011), Hall et al. (2012), Radjenovic et al. (2013) focused on a specific area or a dimension of the fault prediction process. Also, a recent study Kamei and Shihab (2016) has been reported on software fault prediction, but this work has only provided a brief overview of software fault prediction, its components, and discussed some of the accomplishments made in the area. In contrary, our study is focusing on the various dimensions of the software fault prediction process. We identify various activities involved in software fault prediction process, and analyze their influence on the performance of software fault prediction. The review focuses on analyzing the reported works related to these involved activities such as software metrics, fault prediction techniques, data quality issues, and performance evaluation measures. A taxonomical classification of various techniques related to these activities is also presented. It can be helpful in the selection of suitable techniques for performing activities in a given prediction environment. In addition, we have presented the tabular summary of existing works focused on the discussion on advantages and drawbacks in these reviewed work. Statistical study and observations have also been presented.

The rest of the paper is organized as follows. Section 2 gives the overview of the software fault prediction process. In Sect. 3, we present information about the software fault dataset. It includes detail of software metrics, project's fault information, and meta information about the software project. Section 4 has the information of methods used for building software

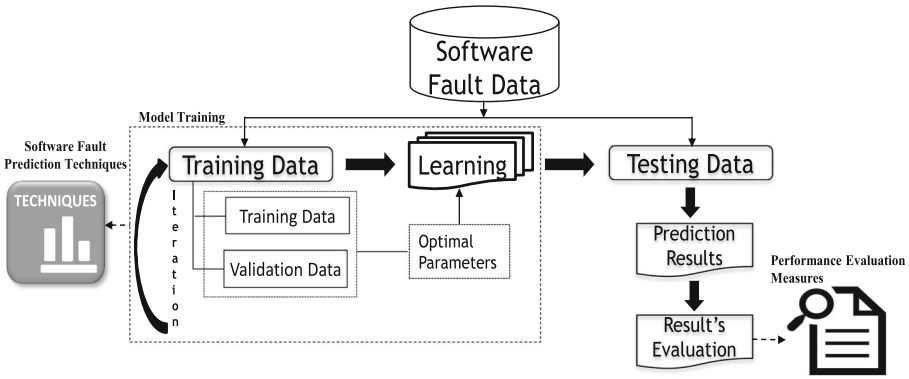

Fig. 1 Software fault prediction process

fault prediction models. Section 5 contained the detail of performance evaluation measures. Section 6 has the results of statistical studied and observations made involving the finding of our review study. Section 7 has the discussion of the presented review work. Section 8 highlighted some key challenges and future works of software fault prediction. Section 9 presented the conclusions.

2 Software fault prediction

Software fault prediction aims to predict fault-prone software modules by using some underlying properties of the software project. It is typically performed by training a prediction model using project properties augmented with fault information for a known project, and subsequently using the prediction model to predict faults for unknown projects. Software fault prediction is based on the understanding that if a project developed in an environment leads to faults, then any module developed in the similar environment with similar project characteristics will ends to be faulty (Jiang et al. 2008). The early detection of faulty modules can be useful to streamline the efforts to be applied in the later phases of software development by better focusing quality assurance efforts to those modules.

Figure 1 gives an overview of the software fault prediction process. It can be seen from the figure that three important components of software fault prediction process are: Software fault dataset, software fault prediction techniques, and performance evaluation measures. First, software fault data is collected from software project repositories containing data related to the development cycle of the software project such as source code and change logs, and the fault information is collected from the corresponding fault repositories. Next, values of various software metrics (e.g., LOC, Cyclomatic Complexity etc.) are extracted, which works as independent variables and the required fault information with respect to the fault prediction (e.g., the number of faults, faulty and non-faulty) work as the dependent variable. Generally, statistical techniques and machine learning techniques are used to build fault prediction models. Finally, the performance of the built fault prediction model is evaluated using different performance evaluation measures such as accuracy, precision, recall, and AUC (Area Under the Curve). In addition to the brief discussion on these four aforementioned components of software fault prediction, upcoming sections present detailed reviews on the various reported works related to each of these components.

3 Software fault dataset

Software fault dataset that act as training dataset and testing dataset during software fault prediction process mainly consists of three component: set of software metrics, fault information like faults per module, and meta information about project. Each of three are reviewed in detail in upcoming subsections.

3.1 Project's fault information

The fault information tells about how faults are reported in a software module, what is the severity of the faults, etc.? In general, three types of fault dataset repositories are available to perform software fault prediction (Radjenovic et al. 2013).

Private/commercial In this type of repository, neither fault dataset nor source code is available. This type of dataset in maintained and used by the companies within the organizational use. The study based on these datasets may not be repeatable.

Partially public/freeware In this type of repository only the project's source code and fault information are available. The metric values are usually not available (Radjenovic et al. 2013). Therefore, it requires that the user must calculate the metric values from the source code and map them to the available fault information. This process requires additional care since calculating metric values and mapping their fault information is a vital task. Any error can lead to the biased learning.

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

The fault data are collected during requirements, design, development, and in various testing phases of the software project and are recorded in a database associated with the software's modules (Jureczko 2011). Based on the phase of the availability of the fault information, faults can be classified as pre-release faults or post-release faults. Sometime dividing faults into separate severity categories can help software engineers to focus their testing efforts on the most sever modules first or to allocate the testing resources optimally (Shanthi and Duraiswamy 2011).

Some of the projects' fault datasets contained information on both number of faults as well as severity of faults. Example of such datasets are KC1, KC2, KC3, PC4, and Eclipse 2.0, 2.1, 3.0 etc. from the PROMISE data repository.

3.2 Software metrics

For an effective and efficient software quality assurance process, developers often need to estimate the quality of the software artifacts currently under development. For this purpose, software metrics have been introduced. By using metrics, a software project can be quantitatively analyzed and its quality can be evaluated. Generally, each software metric is related to some functional properties of the software project such as coupling, cohesion, inheritance, code change, etc., and is used to indicate an external quality attribute such as reliability, testability, or fault-proneness (Bansiya and Davis 2002).

Figure 2 shows the broad classification of software metrics. Software metrics can be grouped into two classes-Product Metrics and Process Metrics. However, these classes are not mutually exclusive and there are some metrics, which act as product metrics as well as process metrics.

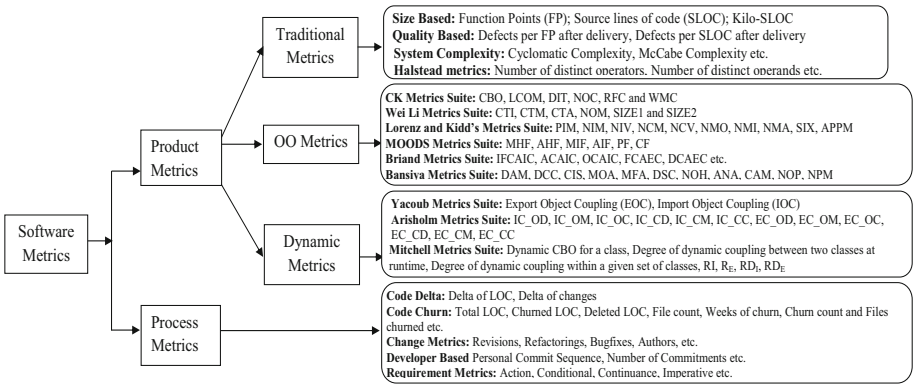

Fig. 2 Taxonomy of software metrics

(a) *Product metrics* Generally, product metrics are calculated using various features of finally developed software product. These metrics are generally used to check whether software product confirms certain norms such as ISO-9126 standard. Broadly, product metrics can be classified as traditional metrics, object-oriented metrics, and dynamic metrics (Bundschuh and Dekkers 2008).

1. *Traditional metrics* Software metrics that were designed during the initial days of emergence of software engineering can be termed as traditional metrics. It mainly includes the following metrics:
   - *Size metrics* Function Points (FP), Source lines of code (SLOC), Kilo-SLOC (KSLOC)
   - *Quality metrics* Defects per FP after delivery, Defects per SLOC (KSLOC) after delivery
   - *System complexity metrics* "Cyclomatic Complexity, McCabe Complexity, and Structural complexity" (McCabe 1976)
   - *Halstead metrics* n1, n2, N1, N2, n, v, N, D, E, B, T (Halstead 1977)
2. *Object-oriented metrics* Object-oriented (OO) metrics are the measurements calculated from the softwares developed using OO methodology. Many OO metrics suites have been proposed to capture the structural properties of a software project. In Chidamber and Kemerer (1994) proposed a software metrics suite for OO software known as CK metrics suite. Later on, several other metrics suites have also been proposed by various researchers such as Harrison and Counsel (1998), Lorenz and Kidd (1994), Briand et al. (1997), Marchesi (1998), Bansiya and Davis (2002), Al Dallal (2013) and others. Some of the OO metrics suites are as follows:
   - *CK metrics suite* "Coupling between Object class CBO), Lack of Cohesion in Methods (LCOM), Depth of Inheritance Tree (DIT), Response for a Class (RFC), Weighted Method Count (WMC) and Number of Children (NOC)" (Chidamber and Kemerer 1994)
   - *MOODS metrics suite* "Method Hiding Factor (MHF), Attribute Hiding Factor (AHF), Method Inheritance Factor (MIF), Attribute Inheritance Factor (AIF), Polymorphism Factor (PF), Coupling Factor (CF)" (Harrison and Counsel 1998)

- *Wei Li and Henry metrics suite* "Coupling Through Inheritance, Coupling Through Message passing (CTM), Coupling Through ADT (Abstract Data Type), Number of local Methods (NOM), SIZE1 and SIZE2" (Li and Henry 1996)
- *Lorenz and Kidd's metrics suite* "PIM, NIM, NIV, NCM, NCV, NMO, NMI, NMA, SIX and APPM" (Lorenz and Kidd 1994)
- *Bansiya metrics suite* "DAM, DCC, CIS, MOA, MFA, DSC, NOH, ANA, CAM, NOP and NOM" (Bansiya and Davis 2002)
- *Briand metrics suite* "IFCAIC, ACAIC, OCAIC, FCAEC, DCAEC, OCAEC, IFCMIC, ACMIC, OCMIC, FCMEC, DCMEC, OCMEC, IFMMIC, AMMIC, OMMIC, FMMEC, DMMEC, OMMEC" (Briand et al. 1997)

3. *Dynamic metrics* Dynamic metrics refer to the set of metrics which depends on the features gathered from a running program. These metrics reveal behavior of the software components during execution, and are used to measure specific runtime properties of programs, components, and systems (Tahir and MacDonell 2012). In contrary to the static metrics that are calculated from static non-executing models. The dynamic metrics are used to identify the objects that are the most run-time coupled and complex objects. These metrics give different indication on the quality of the design (Yacoub et al. 1999). Some of the dynamic metrics suites are given below:

   - *Yacoub metrics suite* "Export Object Coupling (EOC) and Import Object Coupling (IOC)" (Yacoub et al. 1999).
   - *Arisholm metrics suite* "IC_OD, IC_OM, IC_OC, IC_CD, IC_CM, IC_CC, EC_OD, EC_OM, EC_OC, EC_CD, EC_CM, EC_CC" (Arisholm 2004)
   - *Mitchell metrics suite* "Dynamic CBO for a class, Degree of dynamic coupling between two classes at runtime, Degree of dynamic coupling within a given set of classes, R_I, R_E, RD_I, RD_E" (Mitchell and Power 2006)

(b) *Process metrics* Process metrics refer to the set of metrics, which depends on the features collected across the software development life cycle. These metrics are used to make strategic decisions about the software development process. They help to provide a set of process measures that lead to long-term software process improvement (Bundschuh and Dekkers 2008).

We measure the effectiveness of a process by deriving a set of metrics based on outcomes of the process such as- Number of modules changed for a bug-fix, Work products delivered, Calendar time expended, Conformance to the schedule, and Time and effort to complete each activity (Bundschuh and Dekkers 2008).

1. *Code delta metrics* "Delta of LOC, Delta of changes" (Nachiappan et al. 2010)
2. *Code churn metrics* "Total LOC, Churned LOC, Deleted LOC, File count, Weeks of churn, Churn count and Files churned" (Nagappan and Ball 2005)
3. *Change metrics* "Revisions, Refactorings, Bugfixes, Authors, Loc added, Max Loc Added, Ave Loc Added, Loc Deleted, Max Loc Deleted, Ave Loc Deleted, Codechurn, Max Codechurn, Ave Codechurn, Max Changeset, Ave Changeset and Age" (Nachiappan et al. 2010)
4. *Developer based metrics* "Personal Commit Sequence, Number of Commitments, Number of Unique Modules Revised, Number of Lines Revised, Number of Unique Package Revised, Average Number of Faults Injected by Commit, Number of Developers Revising Module and Lines of Code Revised by Developer" (Matsumoto et al. 2010)
5. *Requirement metrics* "Action, Conditional, Continuance, Imperative, Incomplete, Option, Risk level, Source and Weak phrase" (Jiang et al. 2008)

6. *Network metrics* "Betweenness centrality, Closeness centrality, Eigenvector Centrality, Bonacich Power, Structural Holes, Degree centrality and Ego network measure" (Premraj and Herzig 2011)

A lot of work is available in the literature that evaluated the above mentioned software metrics for software fault prediction. In the next sub-section, we have presented a through review of these works and have also summarized our overall observations.

Observations on software metrics

Various work have been performed to analyze the capabilities of the software metrics for software fault prediction. With the availability of the NASA and PROMISE data repositories, the paradigm has been shifted and the researchers started performing their studies using open source software projects (OSS). The benefit of using OSS is that it is easy for anyone to replicate the study and verify the finding of the investigation. We have performed an extensive review of the various studies reported in this direction, as summarized in Table 1. The table summarizes the metrics evaluated, context of evaluation capability of for which evaluation performed, techniques used for evaluation, and advantages and disadvantages of each study. We have drawn some observations from this literature review as discussed below.

- It was found that software developed in the open source environment possesses different characteristics compared to the software developed in the commercial environment (Menzies et al. 2010). So, the metrics performing satisfactory in the one environment may not perform same in the other.
- After 2005, some software metrics suites such as code change metrics, code churn metrics, developer metrics, network metrics, and socio-technical metrics have been proposed by various researchers (Nachiappan et al. 2010; Premraj and Herzig 2011; Jiang et al. 2008; Matsumoto et al. 2010). Some empirical investigations have also been performed by the researchers to evaluate these metrics for fault prediction process. It was found that these metrics have significant correlation with fault proneness (Krishnan et al. 2011; Ostrand et al. 2004, 2005; Premraj and Herzig 2011).
- A lot of studies have evaluated OO metrics (specifically CK metrics suite) for their performance in software fault prediction. Most of the studies confirmed that CBO, WMC, and RFC are the best predictors of faults. Further, most of the work (Li and Henry 1993; Ohlsson et al. 1998; Emam and Melo 1999; Briand et al. 2001; Gyimothy et al. 2005; Zhou and Leung 2006) analyzing LCOM, DIT, and NOC reported that these metrics are having a weak correlation with software fault prediction. Some other OO metrics suites like MOODS and Lorenz and Kidd's are also evaluated by the researchers (Tang et al. 1999; Lorenz and Kidd 1994; Martin 1995). But, more studies are needed to establish the usefulness of these metrics.
- Earlier studies (Hall et al. 2012; Shivaji et al. 2009) showed that the performance of the fault prediction models vary in accordance with the used sets of metrics. However, none of the metrics set was found that always provides the best results regardless of the classifier used.
- Some works (Shivaji et al. 2009; Nagappan et al. 2006; Elish et al. 2011; Rathore and Gupta 2012a) found that combination of metrics from different metrics-suites produced significant results for fault prediction. Like, Shin and Williams (2013) used complexity, code churn, and developer activity metrics for fault proneness prediction and concluded that combination of these metrics produced relatively better results. In another study, Bird et al. (2009) combined the socio-technical metrics and found that a combination of metrics from different sources increased the performance of the fault prediction model.

Table 1 Summarized studies related to software metrics

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Object-oriented metrics						
Li and Henry (1993)	Fault proneness	All CK metrics	Two commercial software	Logistic regression	First study that evaluated CK metrics suite for fault prediction. Results found that all metrics except LCOM accurately predicted fault proneness	Correlation of considered metrics with fault proneness not investigated
Ohlsson et al. (1998)	Fault proneness	Various design metrics	Ericsson Telecom AB system	Principal component and discriminant analysis	Evaluated the fault prediction capability of various design metrics. Found that all used metrics were significantly correlated to fault proneness	The study was performed over only one software project. Evaluation of fault prediction models not thorough
Tang et al. (1999)	Fault proneness	All CK metrics	Three industrial real-time systems	Logistic regression analysis	Investigated the correlation of OO metrics with fault proneness. Found that RFC and WMC metrics were significantly correlated to fault proneness	No fault prediction model has been built to assess the capability of considered metrics for fault prediction
Chidamber et al. (1998)	Productivity and design efforts	All CK metrics	Three commercial trading systems	Stepwise regression	Assessed the usefulness of OO metrics for practicing project management information. Result found that CBO and LCOM were significantly correlated with fault proneness	Only correlation analysis has been used to assess capability of considered metrics. No fault prediction model has been built

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Emam and Melo (1999)	Fault proneness	All Briand metrics	One version of a commercial Java application	Logistic regression	Evaluated the capability of software metrics over the subsequent releases of software project. Found that only OCAEC, ACMIC and OCMEC metrics correlated with fault-proneness	The study was performed over only one software system. Evaluation of fault prediction models not exhaustive
Wong et al. (2000)	Fault proneness	Design metrics	A telecommunication system	Statistical analysis techniques	Presented new design metrics for fault prediction. Results indicated that used metrics were significantly correlated to fault-proneness	The validation of proposed metrics is required using datasets of different domains
Glasberg et al. (1999)	Fault proneness	All CK metrics	One version of a commercial Java application	Logistic regression	Evaluated OO metrics for fault prediction and overall quality estimation. Result found that all considered metrics were correlated to fault proneness	Experimental study not exhaustive. No confusion matrix measure has been used to evaluate the performance of fault prediction model
Briand et al. (2001)	Fault proneness	All metrics of CK metrics suites and Briand metrics suite	An open multi-agent system development environment	Logistic regression and principal component analysis	Explored the relationship between OO design metrics and fault proneness. Results found that coupling metrics are important predictor of faults. The impact of export coupling on fault-proneness is weaker than import coupling	Only correlation measure has been used to evaluate the considered metrics. Only one software project has been used to perform the experiments

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Shanthi and Duraiswamy (2011)	Error proneness	All MOOD metrics	Mozilla e-mail suite	Logistic regression, decision tree and neural network	First study that evaluated MOOD metrics suite for fault prediction. Found that all metrics were significantly correlated with error proneness	Evaluation of fault prediction models not thorough. Correlation of used metrics with fault prediction has not been calculated
Shatnawi et al. (2006)	Error proneness and design efforts	Various OO design metrics	Eclipse 2.0	Univariate binary regression and stepwise regression	Evaluated OO metrics for predicting various quality factors. CTA and CTM metrics are associated with error proneness	Only one software project has been used to perform the experiments. Only statistical measures have been used to evaluate the results
Olague et al. (2007)	Error and fault proneness	All MOOD metrics	Mozilla Rhino	Univariate and Multivariate logistic regression	Evaluated the capability of OO metrics for agile software development processes. Results found that none of the metric was correlated with fault proneness	Only regression analysis has been used to build and evaluate fault prediction model
Shatnawi and Li (2008)	Error proneness	Various OO design metrics	Three release of Eclipse project	Multivariate logistic regression	Evaluated software metrics for fault severity prediction. Found that CTA, CTM and NOA metrics were good predictors of class-error probability in all error severity categories	The fault dataset has been collected using some commercial tools and thus its accuracy is uncertain

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Kpodjedo et al. (2009)	Number of defects	All CK metrics, class rank (CR), evolution cost (EC)	Ten versions of Rhino	Logistic regression, classification regression trees	Proposed two new search-based metrics. Results showed that WMC, LCOM, RFC, and EC metrics were useful for defect prediction	No theoretical validation of proposed metrics has been presented. Metrics extraction was based on homemade tools and no validation of the tool has been provided
Selvarani et al. (2009)	Defect proneness	RFC, WMC, and DIT	Two commercial projects	Property based analysis domain knowledge of experts	Evaluated OO metrics for software fault prediction based on their threshold values. The influence of DIT on defect proneness was 10–33% for a value of 1–3. RFC with value above 100 causes more defects. The value of WMC between 25 and 60 does not cause faults	Only three metrics have been used for the study. No detail of fault datasets has been provided
Elish et al. (2011)	Fault proneness in package level	Martin, MOOD, and CK metrics suites	Eclipse project	Spearman's correlation and multivariate regression	Evaluated three package-level metrics suites for fault prediction. Found that Martin metrics suite is more accurate than the MOOD and CK suites	The validation of work is needed through some more case studies
Singh and Verma (2012)	Fault prediction	All CK metrics	Two version of iText, a JAVA-PDF library software	J48 and Naive Bayes	Evaluated CK metrics suite for fault prediction over some open source projects. Result showed that CK metrics were useful indicator of fault proneness	The fault dataset has been collected using some commercial tools and thus its accuracy is uncertain. The evaluation of results is limited

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Chowdhury and Zulkernine (2011)	Prediction of vulnerability	Complexity, coupling, and cohesion metrics (CCC metrics)	Mozilla Firefox	Decision tree, Naive Bayes, random forests, and logistic regression	Evaluated OO metrics for vulnerability prediction. Found that CCC metrics can be used in vulnerability prediction, irrespective to used prediction technique	Only CCC metrics have been evaluated for vulnerability prediction. Evaluation of other metrics for vulnerability prediction is missing
Dallal and Briand (2010)	Early stage fault prediction	Connectivity-based object oriented class cohesion metrics	Four open-source software projects	Correlation and principal-component analyses, logistic regression	Results indicated that low value of cohesion leads to more faults. Path-connectivity cohesion metrics produced better results than most of the cohesion metrics	If two or more features is of same type, then metric merges such features and it become difficult to tell which attribute is expected to be accessed by a method when methods of same types called
Rathore and Gupta (2012a)	Fault prediction	18 OO metrics	6 Releases of PROP dataset from PROMISE repository	Spearman correlation, logistic, and linear regression	Evaluated various OO metrics individually and in combination with other metrics. Found that classes with high coupling is more likely to fault prone. Cohesion and Inheritance metrics are not relevant to predict fault proneness	More datasets are needed to establish the usefulness of proposed methodology
Rathore and Gupta (2012b)	Fault prediction	18 OO metrics	5 softwares with their 16 releases from PROMISE repository	Spearman correlation, logistic, and linear regression	validated OO metrics for fault prediction over multiple releases of software. Concluded that except cohesion metric, all other considered metrics were significant predictor of faults	More case studies are needed to establish the usefulness of the proposed methodology

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Peng et al. (2015)	Software fault prediction	CK, Martin's, QMOOD, extended CK metrics suites, complexity metrics, and LOC	10 PROMISE project datasets with their 34 releases	J48, logistic regression, Naive Bayes, decision table, SVM, and Bayesian network	Determine a subset of useful metrics for fault prediction. Top five frequently used software metrics produced the fault prediction results comparable to the results produced by using full set of metrics	Fault prediction models built using all, filter and top5 metrics only. Other combinations of software metrics are also possible. Statistical tests have been used without analyzing the distribution of the data
Madeyski and Jureczko (2015)	Defect prediction	CK, Martin, QMOOD, extended CK metrics suites, complexity, LOC, NDC, NR NML, and NDPV	12 open source projects with their 27 releases from PROMISE data repository	Pearsons correlation analysis and hypothesis testing	Evaluated several process metrics for defect prediction. Result showed that NDC and NML metrics have significant correlation with fault proneness. NR and NDPV produced no significant correlation with defect prediction	Only linear regression analysis has been used to build defect prediction model. Other fault prediction can also be used for defect prediction. Evaluation of presented methodology is limited
Process metrics						
Graves et al. (2000)	Predicting number of faults	Change history	A legacy system written in C	Generalized linear models	Evaluated change history and age metrics for fault prediction. Results showed that numbers of changes to the module was the best predictor, while number of developers did not help in predicting numbers of faults	Only one software project has been used for the experiments. No separate fault prediction model has been built to evaluate the software metric
Nikora and Munson (2006)	Fault prediction	Source code metrics, change metrics	Darwin system	Principle component analysis	Define new method for selecting significant metrics for fault prediction. Found that new defined metrics provided high quality fault prediction models	No separate fault prediction model has been built to evaluate the proposed metrics. Comparison analysis not presented

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Hassan (2009)	Prediction of faults	Change complexity metrics,	6 open source projects	Linear regression and statistical test	Proposed complexity metrics based on code change process. Results showed that change complexity metrics were better predictors of fault proneness in comparison to other traditional software metrics	No theoretical validation of proposed metrics has been presented. Only few fault datasets have been used to validate the presented fault prediction methodology
Bird et al. (2009)	Prediction of software failures	Socio-technical networks metrics	Windows vista and the ECLIPSE IDE	Principal component analysis and logistic regression	Investigated the influence of combined socio-technical metrics. Found that socio-technical metrics have produced better recall values than dependency and contribution metrics	Proof of the proposed software metrics is limited. More case studies are required to establish the usefulness of proposed metrics
Nachiappan et al. (2010)	Prediction of defect-prone components	Change bursts suite	Windows Vista	Stepwise regression	Investigated the capabilities of change metrics for fault prediction. Result found that change burst metrics are excellent defect predictors	Evaluation and validation of the results not thorough
Matsumoto et al. (2010)	Fault prediction	Developer metrics suite	Eclipse project dataset	Correlation and linear regression analysis	Studied the effect of developer features on fault prediction. Results showed that developers metrics are good predictors of faults	No theoretical validation of proposed metrics has been presented. Only one software project has been used for experimentation

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Kamei et al. (2011)	Fault prediction	Code clone metrics	Three versions of the Eclipse system (3.0, 3.1 and 3.2)	Logistic regression	Results indicated that relationships between clone metrics and bug density vary with different module sizes and clone metrics showed no improvement for fault prediction	Evaluation on datasets of different domains with other fault prediction techniques requires to validate the presented methodology
Krishnan et al. (2011)	Prediction of fault-prone files	Change metrics suite	Three releases of Eclipse	J48 algorithm	Evaluated change metrics for fault prediction over multiple releases of the software project. Found that all change metrics were good predictors of faults	Only one fault prediction technique has been used to validate the results. Only one dataset with three releases has been used for software fault prediction
Devine et al. (2012)	Faults prediction	Various source code and change metrics	PolyFlow a suite of software testing tools	Spearman correlation	Investigated the association of software faults with other metrics at component level. Found that except average FileChurn and average complexity, all other metrics were positively correlated with faults	More case studies are needed to establish the usefulness of proposed methodology
Ihara et al. (2012)	Bug-fix prediction	41 variables of base, status, period and developer metrics	Eclipse project	regression analysis	Developed a model for bug-fix prediction using various software metrics. Found that the base metrics were the most important metrics to build the model	Only one fault prediction technique has been used to validate the results. Only one software with 3 months of releases has been used to evaluate and validate results

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Rahman and Devanbu (2013)	Defect prediction	14 process metrics, Various code metrics	12 projects developed by Apache	Logistic regression, J48, SVM, and Naive Bayes	Investigated the combined capability of process and code metrics for fault prediction. Found that process metrics always performed significantly better than code metrics	Code metrics have been calculated using some commercial tool, thus the validity of the dataset is uncertain. Evaluation of results not exhaustive
Ma et al. (2014)	Fault proneness	Requirement metrics and design metrics	CM1, PC1	Naive Bayes, AdaBoost, bagging, random forest, logistic regression	First study that evaluated requirement metrics for fault prediction. Results found that combination of requirement and design metrics produced the improved fault prediction results	Only two datasets have been used for the evaluation of considered software metrics. Statistical tests have been used without analyzing the distribution of data
Wu et al. (2014)	Fault prediction	8 developer metrics, 22 process and product metrics	8 open source Java projects	Principle component analysis	Investigated the influence of developer quality on software fault prediction. Found that the combination of developer, process, and product metrics produced improved fault prediction results	Evaluation with more datasets of different domains requires to validate the presented methodology
Xia et al. (2014)	Fault prediction	Code metrics and process metrics	Tracking telemetry and control (TT and C) software	Improved PSO and optimized SVM	Proposed an approach to select useful software metrics for fault prediction. Results showed that out of used code and process metrics, CM, MMSLC, and HM metrics have been found significant for fault prediction	Only two fault datasets has been used to validate the results. The proposed metric selection approach has not been adequately validated to establish its applicability

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Other metrics						
Zhang (2009)	Defect prediction	LOC	Three versions of the Eclipse system (2.0, 2.1 and 3.0), 9 NASA projects	Spearman correlation, multilayer perceptron, logistic regression, Naive Bayes, decision tree	Analyzed the relationship between LOC and software defects. Results showed that a weak but positive relationship exists between LOC and defects. 20% of the largest files are responsible for 62.29% pre-release defects and 60.62% post-release defects	Only one metric (LOC) has been used in the study. Statistical tests have been used without analyzing the domain of the data
Rana et al. (2009)	Number of defects prediction	Software science metrics (SSM)	KC1 dataset	Bayesian, decision tree, linear regression, support vector regression	Found that SSM metrics were effective in classifying software modules as defective or defect free. For number of defects prediction, SSM performance was not up to the mark	A very small dataset has been used to validate the results. Correlation of used metrics has not been evaluated with fault proneness
Mizuno and Hata (2010)	Fault-prone module detection	Complexity and text feature metrics	Three versions of the Eclipse system (2.0, 2.1 and 3.0),	Logistic regression	Evaluated various complexity and text metrics for fault prediction. Found that complexity metrics were better than text metrics for fault prediction	No details of data collection and data preprocessing have been provided. Theoretical validation of proposed metrics is missing

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Nugroho et al. (2010)	Fault proneness	Various UML design metrics and CBO, complexity and LOC	An integrated healthcare system	Univariate and multivariate regression analysis	Proposed and validated various UML metrics for fault prediction. Results showed that ImpCoupling, KSLOC, SDmsg with ImpCoupling and KSLOC metrics were significant predictors of class fault-proneness	Only two aspects of UML diagram have been considered to calculate design metrics. Only one technique has been used to validate the fault prediction model
Arisholm et al. (2010a)	Fault proneness	Various source code and change/ history metrics	A Java legacy system	C4.5, PART, logistic regression, neural network and SVM	Found that LOC and WMC have been significant to predict fault proneness	A single Java software system has been used to validate the results. Optimization of parameter is required to build fault prediction model
Zhou et al. (2010)	Fault proneness	Complexity metrics	Three versions of eclipse	Binary logistic regression	Presented the use of odds ratio for calculating association between software metrics and fault proneness. Showed that LOC and WMC metrics were better fault predictors compare to SDMC and AMC metrics	Only complexity metrics has been used to perform the investigation. No multiple comparison test has been used to determine the difference between the used metrics
Premraj and Herzig (2011)	Defect prediction	Network and code metrics	Open source Java projects, viz., JRuby, ArgoUML and Eclipse	KNN, logistic regression, Naive Bayes recursive partitioning, SVM, tree bagging	Evaluated social-network metrics for fault prediction. Found that network metrics outperformed code metrics for predicting defects. Concluded that using all metrics together did not offer any improvement in prediction accuracy over using network metrics alone	Compared network metrics only with code metrics. Comparison with other software metrics is missing

Table 1 continued

Study	Aim of metric evaluation	Metrics evaluated	Context	Evaluation methods	Advantages	Disadvantages
Ahsan and Wotawa (2011)	Number of bugs prediction	8 Program files logical-coupling metrics	Open source project data from GNOME repository	Stepwise linear regression, PCA and J48	Proposed software metrics using logical-coupling among source files. Found that logical-coupling metrics were significantly correlated with fault prediction	More validation of proposed metrics required to prove their usefulness. More datasets of different domains are required to performed validate the experiment results
Shin and Williams (2013)	Software vulnerabilities	Complexity, code churn, and developer activity metrics (CCD metrics)	Mozilla Firefox web browser and Red Hat enterprise Linux kernel	Logistic regression	Result showed that 24 metrics out of total considered metrics have shown significant discriminant power. A model with subset of CCD metrics can predict vulnerable files	Only one open source project has been used to perform investigation
Stuckman et al. (2013)	Defect prediction	31 source code metrics	19 projects developed by the Apache	Correlation analysis	Evaluated product metrics for software defect prediction. Five class metrics produced small performance gain over LOC metric. Concluded that different metrics appeared significant under the different circumstances	The fault dataset has been collected using some commercial tools and thus its accuracy is uncertain

- Many studies (Zhang 2009; Zhang et al. 2011) have been reported that investigating the correlation between size metric (LOC) and fault proneness. Ostrand et al. (2005) built the model to predict fault density using LOC metrics and found that LOC metric have significant correlation with prediction of fault density. In another study, Zhang (2009) concluded that there is sufficient statistical evidence that a weak but positive relationship exists between LOC and defects. However, Rosenberg (1997) pointed out that there is negative relationship between defect density and LOC. In addition, they concluded that LOC is the most useful feature in fault prediction when combined with other software metrics. In another study, Emam and Melo (1999) demonstrated that there is a simple relationship exist between class size and faults, and there is no threshold effect of class size in the occurrences of faults.
- The use of complexity metrics for building fault prediction model has been examined by various researchers (Li and Henry 1993; Zhou et al. 2010; Olague et al. 2007; Briand et al. 1998). Some of the studies (Zhou et al. 2010; Olague et al. 2007) confirmed the predictive capability of complexity metrics, while others reported the poor performance of these metrics (Binkley and Schach 1998; Tomaszewski et al. 2006). In the study, Olague et al. (2007) reported that the complexity metrics have produced better fault prediction results. Further, it was found that less commonly used metrics like SDMC and AMC are good predictors of fault proneness compared to metrics like LOC and WMC. Zhou et al. (2010) reported that when complexity metrics are used individually, they exhibited the average predictive ability. While, the explanatory power of complexity metrics has increased when they are used with LOC metric.
- Various studies have been performed to evaluate the appropriateness of process metrics for fault proneness (Devine et al. 2012; Moser et al. 2008; Nachiappan et al. 2010; Nagappan and Ball 2005; Nagappan et al. 2006; Radjenovic et al. 2013). Devine et al. (2012) investigated various process metrics and found that most of the process metrics are positively correlated with faults. In another study, Moser et al. (2008) performed a comparative study of various process metrics with code metrics and found that process metrics are able to discriminate between faulty and non-faulty software modules and are better compared to source code metrics. While, Hall et al. (2012) found that process metrics have not performed well compared to OO metrics.

It is observed that there are differences in the results of various studies performed on the set of metrics. Possibly, it is due to the variation in the context in which the data is gathered, the usage of dependent variable (such as fault density, fault proneness, pre-release faults, and post-release faults) during the studies, the implication of linear relationship, and in the performance measures used for evaluation.

3.3 Meta information about project

Meta information about project contained the information of various characteristics (properties) of software project. It consists various set of informations such as the domain of software development, the number of revisions software had, etc. as well as consist information of the quality of the fault dataset used to build fault prediction model. Figure 3 shows various attributes of the meta information about the project.

3.3.1 Contextual information

Context of fault prediction seems to be a key element to establish the usability of the fault prediction models. It is an essential characteristic as in different contexts, fault prediction

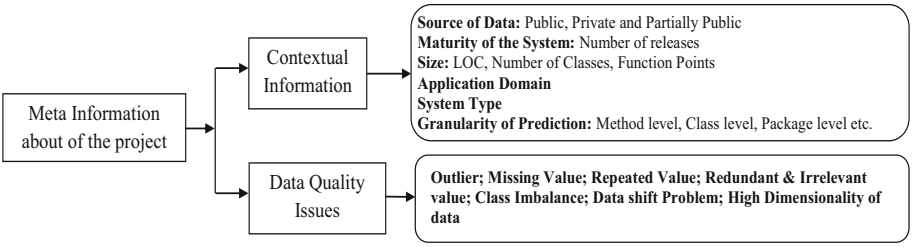

Fig. 3 Meta information of the software project

models may perform differently and the transferability of models between contexts may affect the prediction results (Hall et al. 2012). The current knowledge about the influence of context variables on the performance of fault prediction models is limited. Most of the earlier studies did not pay much attention on the context variables before building the fault prediction model and as a result, selection of a fault prediction model in a particular context is still equivocal. Some of the basic contextual variables/factors that apply to the fault prediction models are given below (Hall et al. 2012):

- *Source of Data* It gives the information about software project dataset over which the study was performed. For example, whether the dataset is from public domain or from the commercial environment. The source of the dataset affects the performance of the fault prediction models. The performance of fault prediction model may varies when transfer to the different datasets.
- *Maturity of the System* Maturity of the system refers to the versions (age) of the software project over which it evolved. Usually, a software project developed over the multiple releases to sustain the changes in the functionality. Maturity of the system has a notable influence on the performance of the fault prediction model. Some model performs better than others do for a new software project.
- *Size* The size of the software project is measure in terms of KLOC (Kilo lines of code). The faults content also varies with the size of the software and it is more likely that the fault prediction model produces different results over the software of different sizes.
- *Application Domain* Application domain indicates the development process and the environment of the software project. Since, different domains use different development practices and it may affect the behaviour of the fault prediction model.
- *The Granularity of Prediction* The unit of code for which prediction has performed known as the granularity of the prediction. It can be faults in a module (class), faults in a file or faults in a package, etc. It is an important parameter since comparing the models having different level of granularity is a difficult task.

Observations on contextual information

The context of fault prediction model has not been comprehensively analyzed in the earlier studies. Some researchers reported the effect of the context over the fault prediction process (Alan and Catal 2009; Calikli et al. 2009; Canfora et al. 2013; Zimmermann et al. 2009), but it is not adequate to make any generalized argument. Hall et al. (2012) analyzed 19 papers in their SLR study related to context variables and found evidence that context variables affect the dependability of fault prediction model. They evaluated the papers in terms of, "the source of data, the maturity of the system, size, application area, and programming language of the system(s) studied". They suggested that it might be intricate to predict faults in some software projects compare to others because they may have a different fault distribution

profile relative to the other software projects. They found that the large sized software projects increase the probability of faults detection compare to small one. In addition to this, they found that maturity of the system has a little or no difference on the model's performances. Also, suggested that there is no relationship between the model performance and the programming language used or the granularity level of prediction.

Calikli et al. (2009) reported that source file level defect prediction improved the verification performance, while decreased the defect prediction performance. Menzies et al. (2011) concluded that instead of looking for the general principles that apply to many projects in empirical software engineering, we should find the best local lessons that are applicable to the groups of similar types of projects. However, Canfora et al. (2013) reported that the multi-objective cross-project prediction outperformed the local fault prediction. The above discussion leads to the conclusion that the context of the fault models has not been adequately investigated and still there is an ambiguity about their use and applicability. It is therefore necessary to perform studies that analyze the effect of various context variables on fault prediction models. This will help researchers to conduct replicated studies and increase the knowledge of the users to select the right set of techniques for the particular context of the problem.

3.3.2 Data quality issues

The quality of fault prediction model depends on the quality of the dataset. It is a crucial step to obtain a software fault dataset with reasonable quality in the fault prediction process. Typically, the fault prediction studies are performed over the datasets available in the public data repositories. However, ease of availability of these datasets can be dangerous as the dataset may be stuffed with unnecessary information that leads to deteriorate the classifier performance. Moreover, most of the studies reported results without any scrutiny of the data and assume that the datasets are of reasonable quality for prediction. There are many quality issues associated with software fault datasets that we need to handle/remove before using them for the prediction (Gray et al. 2011).

- *Outlier* Outliers are the data objects that do not meet with the general behavior of the data. Such data points, which are different from the remaining data are called outlier (Agarwal 2008). Outliers are of particularly important in the fault prediction since outliers may indicate the faulty modules also. Any arbitrary removal of such points may leads to insignificant results.
- *Missing Value* Missing value deals with the values that are left blank in the dataset. Some of the prediction techniques are automatically deal with the missing values and no especial care is required (Gray et al. 2011).
- *Repeated Value* Repeated attributes occur where two attributes have identical values for each instance. This effectively results in a single attribute being over described. For the data cleaning, we remove one of the attributes, so that the values are only being represented once (Gray et al. 2011).
- *Redundant and Irrelevant value* Redundant instances occur when the same features (attributes) describe multiple modules with the same class label. Such data points are problematic in the context of fault prediction, where it is essential that classifiers be tested upon data points independent of those used during training (Gray et al. 2011).
- *Class Imbalance* Class imbalance represents a situation where certain type(s) of instances (called as minor class) are rarely present in the dataset compared to the other types of instances (called as major class). It is a common issue in prediction, where the instances

of major class dominate the data sample as opposed to the instances of the minor class. In such cases, learning of the classifiers may be biased towards the instances of major class. Moreover, classifiers can produce poor results for the minor class instances (Moreno-Torres et al. 2012).

– *Data shift Problem* Data shifting is a problem where the joint distribution of training data is differed from the distribution of testing data. Data shift occurs when the testing (unknown) data experience an event that leads to a change in the distribution of a single feature, a combination of features (Moreno-Torres et al. 2012). It has an inauspicious effect of the performance of the prediction models and needs to be corrected before building any prediction model.
– *High Dimensionality of Data* High dimensionality of the data is a situation where data are stuffed with the unnecessary features. Earlier studies in this regard have confirmed that a high number of features (attributes) may lead to lower classification accuracy and higher misclassification errors (Kehan et al. 2011; Rodriguez et al. 2007). Higher dimensional data can also be a concern for many classification algorithms due to its high computational cost and memory usage.

Observations on data quality issues

Table 2 listed the studies related to data quality issues. Recently, the use of machine-learning techniques in software fault prediction has been increased extensively (Swapna et al. 1997; Guo et al. 2003; Koru and Hongfang 2005; Venkata et al. 2006; Elish and Elish 2008; Catal 2011). But, due to the issues associated with NASA and PROMISE datasets (Gray et al. 2000; Martin 1995), the performance of the learners are not up to the mark. The results of our literature review show that data quality issues have not been investigated adequately. Some of the studies explicitly handled data quality issues (Calikli and Bener 2013; Shivaji et al. 2009), but they are very few (Shown in Table 2). The mostly discussed issues are "high data dimension", "outlier", and "class imbalance" (Rodriguez et al. 2007; Seiffert et al. 2008, 2009; Alan and Catal 2009), while other issues like "data shifting", "missing values", "redundant values" have been ignored by earlier studies.

In the study, Gray et al. (2011) acknowledged the importance of data quality. They highlighted the various data quality issues and presented an approach to redeem them. In another study, Shepperd et al. (2013) analyzed five papers published in the IEEE TSE since 2007 for their effectiveness in handling data quality issues. They found that the previous studies handled data quality isseus insufficiently. They suggested that researchers should specify the source of the datasets they used and must report any preprocessing scheme that helps in meaningful replication of the study. Furthermore, they suggested that researchers should invest efforts in identifying the data characteristics before applying any learning algorithm.

4 Methods to build software fault prediction models

Various techniques for software fault prediction are available in the literature. We have performed an extensive study of the available software fault prediction techniques and based on the analysis of these techniques a taxonomic classification has been proposed, as shown in Fig. 4. Figure shows various schemes that can be used for software fault prediction. A software fault prediction model can be built using the training and testing datasets from the same release of the software project (Intra-release fault prediction), from different releases of the software project (Inter-release fault prediction) or from the different software projects (cross-project fault prediction). Similarly, a software fault prediction model can be used to classify

Table 2 Summarized studies related to data quality issues

Study	Aim	Application project	Methodology used	Advantages	Disadvantages
Noise analysis for fault prediction					
Tang and Khoshgoftaar (2004)	Noise identification	JM1, KC2	K-mean, C4.5	Presented a new technique to determine noisy instances. Concluded that removal of potential noisy instances leads to the improved classification accuracy	No comparison of proposed technique with other noise filtering techniques has been provided
Kim et al. (2011)	Dealing with noise in defect prediction	Columba, Eclipse JDT, Core and Scarab	Bayes Net machine learner, SVM, and Bagging	Proposed an algorithm to detect noisy instances. Found that in general, noises in the defect data do not affect defect prediction performance	Proposed approach was validated using synthetic data. This may not represent the actual noise patterns
Ramler and Himmelbauer (2013)	Impact of noise on defect prediction results	Datasets from different software repositories	Decision tree learner FS-ID3, and a fuzzy-logic decision tree	Investigated various noise-causing factors in defect prediction model. Results indicated that noise level up to 30% does not effect fault prediction performance	Noise causing factors in defect prediction model was not thoroughly studied. Only one technique used to validate the proposed approach
Feature selection techniques for fault prediction					
Rodriguez et al. (2007)	Detecting faulty modules using feature selection	5 Promise software data (KC1, KC2, CM1, PC1 and JM1)	Naive Bayes, C4.5, CFS, CNS, and FCBF	Results showed that smaller datasets with less attributes maintained or improved the prediction accuracy. Wrapper mode was better than the filter mode for feature selection	Empirical evaluation of the results is limited

Table 2 continued

Study	Aim	Application project	Methodology used	Advantages	Disadvantages
Shivaji et al. (2009)	Reduce features to improve bug prediction	8 open source software systems	Naive Bayes, support vector machine, and gain ratio	Presented a new feature selection approach. Results found that reduced features set helps in better and faster bug prediction	No comparison with other feature selection techniques has been provided. More case studies are required to establish the usefulness of proposed approach
Wasikowski and Chen (2010)	Combating the class imbalance problem using feature selection	Microarray and mass spectrometry, NIPS, and character recognition systems	SVM, 1-NN, Naive Bayes, and 7 feature selection techniques	Proposed an approach for handling class imbalance problem. Found that feature selection helps in handling high-dimensional imbalanced data sets	Only few datasets related to fault prediction have been used in the study
Wang et al. (2010b)	Comparative study of threshold-based feature selection techniques	Eclipse project (v3.0)	Naive Bayes, multilayer perceptron, KNN, SVM, and Logistic regression	Selection of important features using threshold values improved the fault prediction performance. Found that OR, GI, and PR-based filters did not perform better compared to other used filters	Evaluation of proposed methodology not exhaustive. Only one dataset has been used to perform the investigation
Xia et al. (2014)	A metric selection approach	PC4 and PC5	Combination of Relief and Linear Correlation	Proposed a new feature selection approach. Found that the proposed feature selection approach improved fault prediction results	More datasets of different domains are required to prove the usefulness of proposed approach. Empirical study not thorough

Table 2 continued

Study	Aim	Application project	Methodology used	Advantages	Disadvantages
Wang et al. (2010a)	Study of filter-based feature ranking techniques	A telecommunications software system	Six filter-based feature ranking technique and 5 classifiers	Results indicated that the choice of a performance metric influence the classification evaluation results. For feature selection, CS performed best, followed by IG. GR performed worst	Only filter-based feature ranking techniques considered for the study. Statistical tests have been used without analyzing the distribution of data
Kehan et al. (2011)	Investigation on feature selection techniques	A telecommunications software system	7 different feature ranking techniques, five different classifiers	A hybrid approach for feature selection has been presented. Reported that different techniques selected different subset of metrics and presented algorithm performed best	No comparison with other feature selection techniques has been presented
Gao et al. (2012)	Prediction of high risk program modules by selecting software measurements	4 releases of LLTS, 4 NASA projects	5 different classifiers, 4 data sampling, wrapper-based, feature ranking techniques	Investigated the impact of data sampling and feature selection for fault prediction. Found that data sampling techniques improved the classification performance. Selection of classifiers affected the prediction results	More case studies are required to establish the usefulness of presented approach

Table 2 continued

Study	Aim	Application project	Methodology used	Advantages	Disadvantages
Gao and Khoshgoftaar (2007)	Proposed a hybrid approach that incorporates data sampling and feature selection	Datasets from large telecommunications software system	SVM, KNN, 7 feature selection techniques including SelectRUSBoost	Proposed a hybrid approach for class imbalance problem. Results showed that for both learners, proposed SeledctRUSBoost technique showed stable performance across various feature ranking techniques	The validation of proposed approach not exhaustive. No comparison has been provided with other similar type of techniques
Satria and Suryana (2014)	Feature selection approach	CM1, KC1, KC3, MC2, PC1, PC2, PC3, and PC4	Genetic algorithm for feature selection	Proposed a new feature selection approach. Found that the proposed feature selection approach significantly improved the fault prediction result	Empirical study not exhaustive. No comparison with other feature selection techniques has been provided
Imbalance data issue for fault prediction					
Seiffert et al. (2008)	Handling imbalanced data	5 NASA project datasets (C12, CM1, PC1, SP1 and SP3)	5 data sampling techniques and 1 Boosting Algorithm, C4.5 and RIPPER	Investigated the effectiveness of data sampling techniques for software fault prediction. Results showed that use of data sampling techniques improved fault prediction performance. AdaBoost produced better performance than data sampling technique	Only two data sampling techniques have been evaluated and also no comparison with other sampling techniques has been provided

Table 2 continued

Study	Aim	Application project	Methodology used	Advantages	Disadvantages
Khoshgoftaar et al. (2010)	Investigating the effect of attribute selection and imbalanced data in defect prediction	3 Eclipse projects (v2.0, v2.1 and v3.0)	KNN, SVM, 6 filter based feature selection techniques, and Random under sampling	Investigated feature selection and data sampling techniques individually and combined. Found that feature selection with sampled data resulted better performance than feature selection with original data	More case studies are required to prove the usefulness of proposed methodology. Only one data sampling technique has been used to perform the investigation
Ma et al. (2012)	Class imbalance problem in defect prediction	9 Promise software datasets	J48, Naive Bayes, and Random under sampling algorithm	A semi-supervised approach has been used for fault prediction. Reported that proposed technique produced improved fault prediction results	Empirical validation needed for generality of proposed approach. No comparative analysis has been provided
Shatnawi (2012)	Improving software fault prediction for imbalanced data	Eclipse IDE (v2.0)	Naive Bayes, Bayes network, nearest neighbor, SMOTE	Presented a new over-sampling approach. Concluded that proposed technique produced better results compared to SMOTE technique	More empirical studies are needed for generality and acceptance of the proposed approach
Wang and Yao (2013)	Investigating the different types of class imbalance learning methods	10 project datasets from PROMISE data repository	Balance and Random under sampling, threshold-moving, SMOTEBoost, and AdaBoostNC	Found that overall BNC performed best for handling class imbalance problem. Balanced random under sampling performed best, and BNC came to the second for PD measure	Validation of proposed approach is needed through more case studies

Table 2 continued

Study	Aim	Application project	Methodology used	Advantages	Disadvantages
Other data quality issues for fault prediction					
Catal and Diri (2008)	Fault prediction using limited data	JM1, KC2, CM1, PC1 datasets	AIRS, YATSI (Yet Another Two Stage Idea), RF and J48 classifiers	Results showed that performance of J48 and RF techniques degraded for unbalanced data. YATSI technique improved the performance of AIRS algorithm for unbalanced datasets	Only few datasets have been used for empirical investigations. Comparative study not exhaustive
Seiffert et al. (2009)	Improving prediction with data sampling and boosting	15 software datasets	5 data sampling techniques RUS, ROS, SM, BSM, and WE and C 4.5	Found that data sampling techniques improved fault prediction performance significantly. Concluded that boosting is the best technique for handling class imbalance problem	No comparison with previous studies has been provided. Statistical test have been used without analyzing distribution of data
Calikli et al. (2009)	Examine the effect of different granularity levels on fault prediction	AR3, AR4 and AR5 datasets, 11 Eclipse datasets	Naive Bayes	Investigated the effect of granularity level on software defect prediction. Results found that file-level defect prediction performed worse than module-level defect prediction	Not all the granularity level cases considered. Evaluation of the results not thorough. No comparative analysis has been presented

Table 2 continued

Study	Aim	Application project	Methodology used	Advantages	Disadvantages
Alan and Catal (2009)	Outlier detection	jEdit text editor project	Random forests	Found that thresholds based outlier detection is an effective pre-processing step for efficient fault prediction	No concrete results have been presented to prove the effectiveness of proposed approach
Nguyen et al. (2010)	Study of bias in bug fix datasets	IBM Jazz software project	Fishers exact test and a two-sample Kolmogorov Smirnov test	Results showed that even with a near-ideal dataset, biases do exist. Concluded that biases are more likely a symptom of underlying software development process instead of due to the used heuristics	No investigation has been performed for open source datasets. No comprehensive results presented
Gray et al. (2011)	Analyze the quality of NASA datasets	13 NASA datasets	Data cleaning techniques	Concluded that it is important to understand the structural properties of the dataset before building any fault prediction model	Only theoretical validation presented. No empirical analysis has been performed
Verma and Gupta (2012)	Improve fault prediction using two level data pre-processing	5 NASA datasets (CM1, JM1, KC1, PC1 and KC2)	IB1, IBK, K-star and LWL; and Attribute selection and Instance Filtering	Investigated the influence of pre-processing techniques on software fault prediction. Found that two level pre-processing produces better results compare to using them individually	Empirical investigation not exhaustive. Only two pre-processing techniques have been used for the investigation

Table 2 continued

Study	Aim	Application project	Methodology used	Advantages	Disadvantages
Armah et al. (2013)	Investigating the effect of multi-level data preprocessing in defect prediction	5 NASA project datasets	Double attribute selection and triple resampling methods	Found that multi-level preprocessing produced better results compared to using attribute selection and instance independently	The proposed approach not properly validated. No comparative analysis provided
Calikli and Bener (2013)	Handling missing data problem	Four different project datasets	ROWEIS' EM Algorithm	Results showed that EM algorithm produced significant fault prediction results. The results were comparable with the results obtained by using complete data	More studies using different datasets are required to establish the significant of proposed approach. No comparative analysis presented
Shepperd et al. (2013)	Assessing the quality of NASA Software defect datasets	14 NASA datasets	Data preprocessing and cleaning techniques	Recommended that researchers should report details of preprocessing for fault prediction. Also, researchers must analyze domain of the dataset before applying any prediction technique	Only theoretical evaluation presented. Only NASA defect datasets have been used for the analysis
Rodriguez et al. (2012)	Analyzing software engineering repositories and their problems	8 different software fault data repositories	Data cleaning and preprocessing techniques	Discussed various issues related to software fault data repositories. Provided the detail of various available software repositories	The study can be used as reference but did not provide any redeem of the stated problems

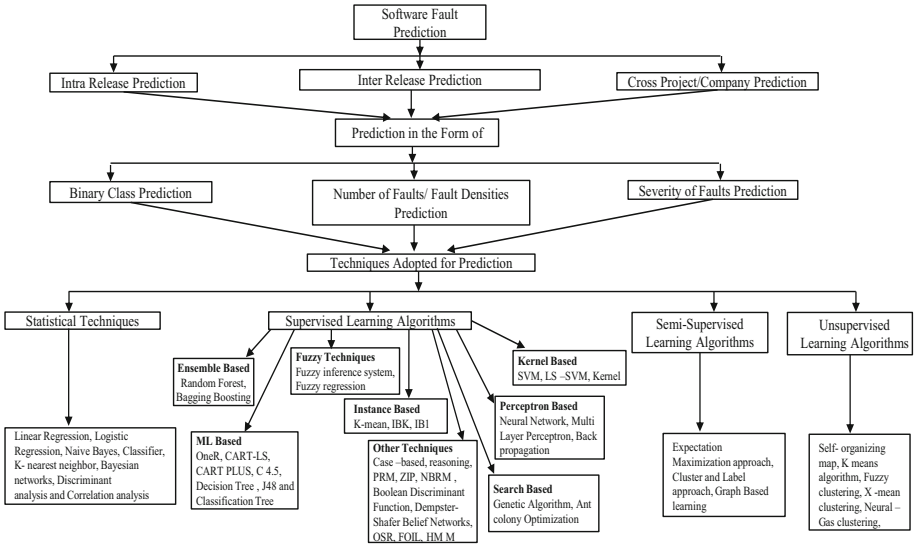

Fig. 4 Taxonomy of software fault prediction techniques

software modules into faulty or non-faulty categories (binary class classification), to predict the number of faults in a software module, or to predict the severity of the faults. Various machine learning and statistical techniques can be used to build software fault prediction models. The different categories of software fault prediction techniques shown in Fig. 4 are discussed in the upcoming subsections.

Three types of schemes can be possible for software fault prediction:

Intra-release prediction Intra release refers to a scheme of prediction in which training dataset and testing dataset both are drawn from the same release or version of the software project. The dataset is divided into training and testing part and cross-validation is used to train the model as well as to perform prediction. In n-folds cross-validation scheme, each time $(n - 1)$ parts are used to train the classifier and rest one part is used for testing. This procedure is repeated n times and the validation results are averaged over the rounds.

Inter-release prediction In this type of prediction, training dataset and testing dataset are drawn from different releases of the software project. The previous release(s) of the software are used as training dataset and the current release is used as testing dataset. It is advised to use this scheme for fault prediction, as the effectiveness of the fault prediction model can be evaluated for the unknown software project's release.

Cross-project/company prediction The earlier prediction schemes make the use of historical fault dataset to train the prediction model. Sometimes, a situation can arise that the training dataset does not exist, because either a company had not recorded any fault data or it is the first release of the software project, for which no historical data is available. In this situation, for fault prediction, analysts would train the prediction model from another project's fault data and use it to predict faults in their project, and this concept is named as cross-project defect prediction (Zimmermann et al. 2009). However, there have been only little evidence that fault prediction works across projects (Peters et al. 2013). Some researchers reported their studies in this regards, but the prediction accuracy was very low with high misclassification rate.

Generally, a prediction model is used to predict the fault-proneness of software modules in one of the three categories: Binary class classification of faults, number of faults/fault density prediction, and severity of fault prediction. A review of various techniques for each of the category of fault prediction discussed in the upcoming subsections.

4.1 Binary class prediction

In this type of fault prediction scheme software modules are classified into faulty or non-faulty classes. Generally, modules having one or more faults marked as faulty and modules having zero faults marked as non-faulty. This is the most frequently used types of prediction scheme. Most of the earlier studies related to fault prediction are based on this scheme such as Swapna et al. (1997), Ohlsson et al. (1998), Menzies et al. (2004), Koru and Hongfang (2005), Gyimothy et al. (2005), Venkata et al. (2006), Li and Reformat (2007), Kanmani et al. (2007), Zhang et al. (2007, 2011), Huihua et al. (2011), Vandecruys et al. (2008), Mendes-Moreira et al. (2012). A number of researchers have used various classification techniques to build the fault prediction model including statistical techniques such as Logistic Regression, Naive Bayes, supervised techniques such as Decision Tree, SVM, Ensemble Methods, semi-supervised techniques such as expectation maximization (EM), and unsupervised techniques such as K-means clustering and Fuzzy clustering.

Different works available in the literature on the techniques for binary class classification are summarized in Table 3. It is clear from the table that for binary class classification, most of the studies have used statistical and supervised learning techniques.

Observations on binary class classification of faults

Various researchers built and evaluated fault prediction models using a large number of classification techniques in the context of binary class classification. Still, it is difficult to make any general arguments to establish the usability of these techniques. All techniques produced an average accuracy between 70% and 85% (approx.) with lower recall values. Overall, it was found that despite some differences in the studies, no single fault classification technique proved superior to the other techniques across different datasets. One reason for the lower accuracy and recall value is that most of the studies have used fault prediction techniques as black box, without analyzing the domain of the datasets and suitability of techniques for the given datasets. The availability of the data mining tools like Weka also worsens the situation. Techniques such as Naive Bayes and Logistic Regression seem to perform better because of their simplicity and easiness for the given dataset. While, techniques such as Support Vector Machine produced poor results due to the complexity of finding the optimal parameters for fault prediction models (Hall et al. 2012; Venkata et al. 2006). Some of the researchers have performed studies comparing an extensive set of techniques for software fault prediction and have reported that the performance of techniques vary with the used dataset and none of the technique always performed the best (Venkata et al. 2006; Yan et al. 2007; Sun et al. 2012; Rathore and Kumar 2016a, c). Most of the studies evaluated their prediction models through different evaluation measures, thus, it is not easy to draw any generalized arguments out of them. Moreover, some issues such as skewed dataset and noisy dataset have not been properly taken care of before building fault prediction models.

4.2 Prediction of the number of faults and fault density

There have been few efforts analyzing fault proneness of the software projects by predicting the fault density or the number of faults in given software such as Graves et al. (2000), Ostrand

Table 3 Works on binary class classification

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Swapna et al. (1997)	Regression tree, density modeling techniques	Accuracy, type I and type II errors	Medical imaging system (MIS)	Assessed the capabilities of regression for software defect prediction. Found that regression technique produced better fault prediction results	Only one dataset has been used to perform the investigation. Empirical study not thorough
Ohlsson et al. (1998)	PCA and discriminant analysis (DA)	Correlation point	A Ericsson Telecom system	Results showed that discriminant coordinates increased with the ordering of modules, thus improving prediction and prioritization efforts	Only statistical techniques have been used for investigation. No concrete results presented
Menzies et al. (2004)	Naive Bayes and J48	Accuracy, PD, precision and PF	CM1, JM1, PC1, KC1, and KC2	Concluded that Naive Bayes produced better results compared to J48. Suggested the use of fault prediction in addition to inspection for better quality assurance activity	No comprehensive empirical study provided. More case studies are required to support the conclusions of the study
Koru and Hongfang (2005)	J48 and K-star	F-measure, precision, and recall	CM1, JM1, KC1, KC2, and PC1	Suggested the use of large datasets for defect prediction. Also showed that defect prediction produced better result with class-level metrics	Investigated the influence of dataset's size on fault prediction only. Other characteristics of the dataset not evaluated

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Challagulla et al. (2005)	Regression (Linear, Logistic, Pace, and SVM), neural network for continuous and discrete goal field, Naive Bayes, IB, J48, and 1-Rule	Mean absolute error	CM1, JM1, KC1, and PC1	Results found that combination of 1-R and instance-based Learning produced better prediction accuracy. Also, found that size and complexity metrics are not sufficient for efficient fault prediction	Built fault prediction models not thoroughly evaluated
Gyimothy et al. (2005)	Logistic regression, decision tree, and neural network	Precision, correctness, and completeness	Mozilla 1.0 to Mozilla 1.6	Presented a toolset to calculate the OO metrics. Results showed that the presented approach did not improved the precision value	The tool is specific to C++ projects. Validation of tool not presented
Venkata et al. (2006)	Memory based reasoning (MBR) technique	Accuracy, PD, and PF	CM1, JM1, KC1, and PC1	Found that for accuracy, simple MBR with euclidian distance perform better than other used techniques. Proposed a framework to derive the optimal configuration for given defect dataset	Empirical study not thorough. No comparative analysis presented

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Yan et al. (2007)	Logistic regression, discriminant analysis, decision tree, rule set, boosting, kernel density, Naive Bayes, J48, IBK, IB1, Voted perceptron, VF1, Hyper Pipes, ROCKY, and Random Forest	PD, Accuracy, Precision, G-mean1, G-mean2, and F- measure	CM1, JM1, KC1, KC2, and PC1	Proposed a novel methodology based on variants of the random forest algorithm for robust fault prediction. Found that overall random forest performed better than all the other used fault prediction techniques	More datasets of different domains are required to prove the usefulness of presented methodology
Menzies et al. (2007)	Naive Bayes, J48, and log filtering techniques	Recall and probability of false alarm	8 NASA datasets	Found that use of static code metrics for defect prediction techniques is useful. Concluded that used predictors were useful for prioritizing of code that need to be inspected	Only few data mining techniques have been used to perform experiments. No concrete results presented to support the proposed methodology
Kanmani et al. (2007)	Back propagation and probabilistic neural network, discriminant analysis and logistic regression	Type I, type II and overall misclassification Rate	PC1, PC2, PC3, PC4, PC5, and PC6	Results showed that probabilistic neural networks outperformed back propagation neural networks in predicting the fault proneness in OO software	The empirical evaluation is limited and only one homemade project has been used for experiments. Comparative study not thorough

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Li and Reformat (2007)	Support vector machine, C4.5, multilayer perceptron and Naive Bayes classifiers	Sensitivity, specificity, and accuracy	JM1 and KC1	A new method has been proposed to perform fault prediction for skewed datasets. Performance of proposed methodology was found better compared to conventional techniques	Feasibility of the proposal is not checked in a concrete manner. Comparative analysis with other techniques limited
Catal and Diri (2007)	Natural immune systems, artificial immune systems and AIRS	G-mean1, G-mean2 and F-measure	JM1, KC1, PC1, KC2 and CM1	Proposed a new fault prediction model. Found that proposed algorithm performed better compared to other used algorithms with class-level data	The proposed fault prediction model only evaluated for OO system. Full automation of proposed model is not available yet
Seliya and Khoshgoftaar (2007)	Expectation maximization, C4.5	Type I, type II and over-all error rate	KC1, KC2, KC3 and JM1	Investigated the use of semi-supervised techniques for fault prediction. Results showed that EM technique improved fault prediction performance	Only one semi-supervised approach used in the study. More empirical investigations are needed for generality and acceptance of the approach
Jiang et al. (2008)	Naive Bayes, Logistic, IB1, J48, Bagging	Various evaluation techniques and cost curve	8 NASA datasets	Introduced the use of cost curve for evaluating fault prediction models. Suggested that selection of best prediction model influence by software cost characteristics	The proposed methodology is still in progress and needs validation through case studies

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Vandecruys et al. (2008)	Ant Miner+, C4.5, logistic regression and support vector machine	Accuracy, specificity and sensitivity	KC1, PC1 and PC4	Investigated the use of search-based approach for fault prediction. Reported that proposed approach is superior than other used approach for fault prediction	The proposed approach did not significantly improve the fault prediction results. Empirical evaluation of the approach is limited
Catal et al. (2009)	X-means clustering, fuzzy clustering, K-means	12 different performance measures	AR3, AR4, and AR5	Presented an unsupervised approach for the fault prediction when historic dataset is not available	No detail about the threshold value selection of software metrics is provided in the paper
Turhan and Bener (2009)	Naive Bayes	recall and probability of false alarm	CM1, KC3, KC4, MW1, PC1, PC2, PC3, and PC4	Found that Naive Bayes with PCA produced improved results. Results showed that assigning weights to static code attribute increased the fault prediction performance	No comparison of Naive Bayes with other techniques presented
Zimmermann et al. (2009)	Logistic regression and decision tree	Accuracy, precision, and recall	Various open source systems	Investigated the effectiveness of cross-project defect prediction models. Concluded that success rate of cross-project prediction is very low	Approach of relevant software metric selection not clearly stated. Accuracy of fault datasets used for the analysis is uncertain

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Pandey and Goyal (2010)	Decision tree and fuzzy rules	Accuracy	KC2 dataset	Proposed a hybrid approach for the prediction of fault-prone software modules. The proposed approach also predicts the degree of fault-proneness of the modules	Validation of results is limited to one software project. Usefulness of the approach compared to existing approaches is not presented in study
Lu et al. (2012)	Random forest and fitting-the fits (FTF)	Probability of detection and AUC	JM1, KC1, PC1, PC3 and PC4	Investigated the use of an iterative semi-supervised approach for fault prediction. Semi-supervised technique outperformed used supervised technique	No concrete results presented to support the generality and acceptance of the model
Bishnu and Bhattacherjee (2012)	K-means, Catal's two stage approach (CT) single stage approach (CS), Naive Bayes and linear discriminant analysis	False positive rate, false negative Rate and Error	AR3, AR4, AR5, SYD1 and SYD2	Evaluated the use of unsupervised approach for fault prediction. Reported that overall error rate of QDK algorithm was comparable to other used techniques	Proposed approach did not significantly improve fault prediction results. No comparison with other fault prediction techniques provided

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Sun et al. (2012)	Naive Bayes with log-filter, J48, and 1R	ROC curve, PD, PF and DIFF	13 datasets from NASA and 4 datasets from the PROMISE repository	Found that different learning schemes must be chosen for different datasets. Results revealed that overall, Naive Bayes performs better than J48, and J48 is better than 1R	Implementation of the proposed approach is missing
Menzies et al. (2011)	WHICH and WHERE cluster	Mann–Whitney test	CHINA, NasaCoc, Lucene 2.4, Xalan 2.6	Found that for empirical SE, instead of using generalized methods, we should use the best local practices for the groups of related projects	Empirical investigations over the datasets of different domains are required for generality of the proposed approach
Shin et al. (2011)	Logistic regression	Recall, precision, and PF	Mozilla Firefox web browser data	Concluded that fault prediction models can also be used for vulnerability prediction. The proposed fault prediction techniques provided higher recall and low precision values	Fault dataset collected from a software system that has short period between releases. Therefore, only few new vulnerability has been introduced in the subsequent releases
Euyseok (2012)	Random forest, multi-layer perceptron (MLP), and SVM	Type I and type II errors	Two training and two testing datasets	RF model significantly outperformed SVM and MLP models	Empirical investigation is limited. No confusion matrix parameters used for results evaluation

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Chatterjee et al. (2012)	Nonlinear autoregressive with eXogenous inputs (NARX) network	Least square estimation, MAE, RMSE and forecast variance	J1, CDS datasets	Found that fault detection process is dependent not only on residual fault content but also on testing time. Reported that proposed technique performed better compared to other used techniques	Applicability of the proposed approach not thoroughly investigated for fault prediction
Sun et al. (2012)	Ensemble of classifiers, three coding schemes and six traditional imbalance data	AUC, statistical test	14 NASA defect datasets	Results showed that proposed method outperformed other sampling techniques. Ensemble of classifiers improved the prediction performance	More case studies are required to generalize the applicability of proposed approach
Lu and Cukic (2012)	Semi-supervised algorithm (FTcF) and ANOVA	AUC and PD	KC1, PC1, PC3 and PC4	Results showed that self training improved performance of supervised algorithms. Also, found that semi-supervised learning algorithms with preprocessing techniques produced improved results	No comparison with other semi-supervised techniques presented

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Li et al. (2012)	Semi-supervised learning method ACoForest	F-measure	4 Eclipse, Lucene, and Xalan datasets	Results showed that sampling with semi-supervised learning performed better than sampling with conventional machine learning techniques	Only one sampling technique used to performed experiments. Evaluation of results is limited to only one evaluation measure
Zhimin et al. (2012)	Naive Bayes, C4.5, SVM, decision table and Logistic regression	precision, recall, and F-measure	10 open source projects with their 34 different releases	Found that cross-project defect predictions influenced by the distributional characteristics of data. In best cases, training data from other projects can provide better prediction results than training data from the same project	Validation of results with more metrics such as code change history, other process metrics are required. Selection of training dataset for fault prediction is not clear
Ma et al. (2012)	Transfer Naive Bayes (TNB)	Recall, precision, AUC and F-measure	7 NASA datasets and 3 SOFTLAB datasets	Results showed that TNB produced more accurate results in terms of AUC, within less runtime than the state of the art methods	Significance of software metrics not evaluated for used fault prediction technique. It may affect the assumptions of the TNB technique
Rahman and Devanbu (2013)	Logistic regression	Accuracy, precision, recall, F-measure, ROC, and cost effectiveness	9 open source software systems	Results suggested that code metrics and process metrics help to improve fault prediction performance	A commercial tool has been used to calculate the metrics. Thus, its accuracy is unreliable

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Canfora et al. (2013)	Multi-objective logistic regression and a clustering approach	Precision and recall	10 projects fault data from PROMISE data repository	Cross-project prediction achieved lower precision than within project prediction. Multi-objective cross-project prediction produced better results	Validation of proposed approach with the datasets of different domains is required for generalization of the approach
Herbold (2013)	EM-clustering and K-nearest neighbor, logistic regression, Naive Bayes, Bayesian network, SVM, and C4.5	Precision, recall, and success rate	44 versions of 14 software projects from PROMISE data repository	Found that training data selection improved the success rate of cross-project prediction significantly. Overall, cross-project prediction produced lower accuracy	Few characteristics of the fault dataset have been considered in the study. Empirical validation considering different fault dataset characteristics is required
Dejaeger et al. (2013)	15 different Bayesian network (BN) classifiers	AUC and H-measure	JM1, KC1, MC1, PC1, PC2, PC3, PC4, PC5 and three versions of Eclipse	Results showed that augmented Naive Bayes classifiers and random forest produced the best results for fault prediction	Need to evaluate proposed h-measure through more case studies
Peters et al. (2013)	Random forest, Naive Bayes, logistic regression, and K-NN; Peters filter and Buraks filter	Accuracy, PF, F-measure and G-measure	56 static code defect data-sets from the PROMISE	Proposed a new technique for cross-company fault prediction. Peter filter worked better than Burak filter and Naive Bayes produced the best results	More empirical validations needed for generality and acceptance of the proposed technique

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Couto et al. (2014)	Granger causality test	Precision, recall, and F-measure	Eclipse JDT, PDE, Equinox framework and Lucene	The proposed fault prediction technique produced precision greater than 50% for three cases out of four systems considered	The parameter values used in the proposed model need to be optimized
Malhotra and Jain (2012)	LR, ANN, SVM, DT, cascade correlation network, Group methods of data handling and gene expression programming	Sensitivity, specificity, AUC, and proportion correct	AR1 and AR6	Evaluated various machine learning and statistical techniques for fault prediction. Results showed that machine-learning techniques outperformed statistical techniques for fault prediction	Only source code metrics have been used to build fault prediction model. Validation with datasets of different domains is required to generalize the results
Park and Hong (2014)	X-means and EM clustering techniques	Accuracy, FPR, FNR, and total error rate	AR3, AR4, and AR5	Proposed an unsupervised technique for fault prediction. Results showed that X-means clustering techniques produced better results compared to other used techniques	Validation of proposed technique was limited. Comparative analysis was not thorough
Panichella et al. (2014)	LR, Bayes network, Radical basis function network, multilayer perceptron, decision tree, and decision table	precision, recall, and AUC	Ant, Jedit	different fault prediction techniques produced different results. No single technique outperformed all other techniques	Other fault prediction techniques such as, local models can be used to validate the results

Table 3 continued

Study	Classifiers used	Performance evaluation measures	Dataset used	Advantages	Disadvantages
Caglayan et al. (2015)	Naive Bayes, logistic regression and Bayes network	Precision, recall and accuracy	An industrial software	Results showed that proposed technique produced higher accuracy in predicting the overall defect proneness	Statistical tests have been used without analyzing distribution of data. Only code metrics used in building fault prediction models
Erturk and Sezer (2015)	Adaptive neuro fuzzy interface system (ANFIS), ANN, and SVM	ROC curve	24 different datasets from the PROMISE data repository	A fuzzy system based technique presented for fault prediction. ANFIS outperformed all other techniques used for fault prediction	The presented approach uses experts to collect the initial fault information. Choosing the right set of expert is critical for the success of the approach

et al. (2005), Janes et al. (2006), Ganesh and Dugan (2007), Rathore and Kumar (2015b), Rathore and Kumar (2015a). Table 4 summarized the studies related to the number of faults prediction. From table it is clear that for the number of faults prediction, most of the studies have used regression based approaches.

Observations on the prediction of number of faults and fault density

Initially, Graves et al. (2000) presented an approach for the number of faults prediction in the software modules. Fault prediction model has been built using a generalized linear regression model and using software change history metrics. The results found that the prediction of number of faults provides more useful information rather than predicting modules being faulty and non-faulty. Later, Ostrand et al. reported a number of studies predicting number of faults in a given file based on LOC, Age, and Program type software metrics (Ostrand et al. 2005, 2004, 2006; Weyuker et al. 2007). They proposed a negative binomial regression based approach for the number of faults prediction and argued that top 20% of the files are responsible for 80% (approx.) of the faults and reported results in this context. Later, Gao and Khoshgoftaar (2007) reported a study, investigating the effectiveness of the count models for fault prediction over a full-scale industrial software project. They concluded that among the different count models used, the zero-inflated negative binomial and the hurdle negative binomial regression based fault prediction models produced better results for fault prediction. These studies reported some results to predict fault density, but they did not provide enough logistics that can prove the significance of the count models for fault density prediction. As well as the selection of a count model for an optimal performance is still equivocal. Moreover, the earlier studies made the use of some change history and LOC metric without providing any appropriateness of these metrics. One more issue is that the quality of fault prediction models were evaluated by using some hypothesis testing and goodness of fit parameters. Therefore, it is difficult to compare these studies on common comparative measure to draw any generalized arguments.

4.3 Severity of faults prediction

Different studies related to the prediction of severity of fault have been summarized in Table 5. Only few works are available in the literature related to the fault severity prediction such as Zhou and Leung (2006), Shatnawi and Li (2008), Sandhu et al. (2011). First comprehensive study for severity of fault prediction has been presented by Shatnawi and Li (2008). They performed the study for Eclipse software project and found that object-oriented metrics are good predictor of fault severity in software project. Later, some other researchers also predicted the fault severity, but they are very few and do not lead to any generalized conclusion. One problem with fault severity prediction is the availability of the datasets. Since, the severity of the fault is a subjective issue and different organizations classified faults into different severity categories accordingly.

5 Performance evaluation measures

Various performance evaluation measures have been reported in the literature to evaluate the effectiveness of fault prediction models. In a broad way, evaluation measures can be classified into two categories: Numeric measures and Graphical measures. Figure 5 illustrates taxonomy of performance evaluation measures. Numeric performance evaluation measures mainly include accuracy, specificity, f-measure, G-means, false negative rate, false positive rate, pre-

Table 4 Works related to number of fault/fault density prediction

Study	Classifiers used	Performance evaluation parameters	Dataset used	Advantages	Disadvantages
Graves et al. (2000)	Generalized linear models (GLM)	Fault contain in modules	IMR database	Explored the use of change metrics for the distribution of faults. Results showed that GLM performed best with numbers of changes to the module metric	Validation and evaluation of the results are limited
Xu et al. (2000)	Fuzzy non-linear regression technique, neural network	Average absolute error	A large telecommunication system	The presented system predicts fault ranges in the software modules	The evaluation of results is limited to one performance measure only. Comparison with other fault prediction approaches is not presented in the paper
Ostrand et al. (2004)	Negative binomial regression	Fault contain in top 20% of files	An inventory tracking system	Proposed an approach for the number of faults prediction. Found that proposed approach was significant for fault prediction. Also, found that file's age and size metrics influenced the fault prediction performance	Evaluated fault prediction models using faults found in top 20% of files only. Validation of results is required using some other evaluation measures also
Venkata et al. (2006)	Decision trees, Naive Bayes, logistic regression, nearest neighbor, 1-Rule, regression and neural network	Mean absolute error	JM1, KC1, PC1, and CM1	Performed a comparative analysis of different techniques for the number of faults prediction. Suggested that prediction of the actual number of faults in a module is much more difficult than predicting it as fault-prone	Results have been evaluated using error measure only. No comparative evaluation of used techniques has been presented

Table 4 continued

Study	Classifiers used	Performance evaluation parameters	Dataset used	Advantages	Disadvantages
Ostrand et al. (2005)	Negative binomial regression	Fault contain in top 20% of files	An inventory system and a provisioning system	A regression model has been proposed to predict the number of faults. Results showed that proposed approach provided significant results for fault prediction	Only two software systems have been used for empirical investigation. Validation of the results is limited
Janes et al. (2006)	Poisson, negative binomial, and zero-inflated negative binomial regression	Correlation coefficient and Alberg diagrams	A telecommunication system	Evaluated the correlation of design software metrics for the number of defects prediction. ZINBR model with RFC provided the best prediction results	Only few design metrics included in the study. No comparative evaluation is presented
Bibi et al. (2006)	Regression via classification (RvC)	Mean absolute error and accuracy	PEKKA Forselious	presented the use of RvC technique for the number of defects prediction. Found that used fault prediction technique produced improved fault prediction results	Evaluation of the results is limited. The validation of results is not generalized
Ganesh and Dugan (2007)	Bayesian classifier, linear, poisson, and logistic regression	Specificity, sensitivity, precision, FPR, and FNR	KC1	Investigated the relation of metrics with fault occurrences. Results showed that proposed approach produced significant results for fault prediction	The findings of the study are difficult to generalize

Table 4 continued

Study	Classifiers used	Performance evaluation parameters	Dataset used	Advantages	Disadvantages
Li and Reformat (2007)	Fuzzy logic and SimBoost	Sensitivity, specificity, and accuracy	JM1 and PC1	Proposed a new method for fault prediction. Found that skewed data is an issue for lower prediction accuracy. The proposed approach provided significant results	Evaluation of the results is not exhaustive. No comparative analysis presented
Gao and Khoshgoftaar (2007)	5 count models	Pearson's Chi-Square, absolute, and relative errors	2 large Windows systems	Presented the use of count models for the number of faults prediction. Found that PRM and HP2 produced poor results compared to other six used methods	The study was performed over only one software system. Evaluation of the results is limited
Shin et al. (2009)	Spearman rank correlation, Pearson correlation, Negative binomial regression	Correlation coefficient, % of faults	A business maintenance system that has had 35 releases	Investigated the influence of calling structure on fault prediction. Results showed that addition of calling structure information provided only marginal improvement in fault prediction accuracy	Theoretical validation of proposed approach is missing. Validation of the results is not exhaustive
Cruz and Ochimizu (2009)	Logistic regression	Hosmer–Lemeshow test	Mylyn, BNS, and ECS datasets	Analyzed the influenced of data transformation on software fault prediction. Found that log transformations increased the accuracy	Findings of the results are limited. Evaluation with the different datasets is required to generalize the findings

Table 4 continued

Study	Classifiers used	Performance evaluation parameters	Dataset used	Advantages	Disadvantages
Ostrand et al. (2010)	Negative binomial regression	Alberg diagram, Fault contain in top 20% of files	3 large industrial software systems	Analyzed the influence of developers over fault prediction. Found that developer information can improve prediction results, but only by a small amount. Also, found that individual developers past performance is not an effective predictor	Evaluation of the results using other performance measures required to establish the finding of the results
Yan et al. (2010)	Fuzzy support vector regression	Mean square error and squared correlation coefficient	Medical imaging system and redundant strapped-down unit projects	Proposed a novel fuzzy based approach for defect numbers prediction. Evaluated the results for two commercial datasets	Evaluation of results is not thorough. Validation of the approach with some larger datasets is required
Liguo (2012)	Negative binomial regression (NBR) and Binary logistic regression (BLR)	Accuracy, precision, and recall	6 releases of Apache Ant	Results showed that NBR is not as effective as BLR in predicting fault-prone modules. Also, found that NBR is effective in predicting multiple bugs in one module	Evaluation of the results is limited. Comparative analysis is not thorough

Table 4 continued

Study	Classifiers used	Performance evaluation parameters	Dataset used	Advantages	Disadvantages
Yadav and Yadav (2015)	Fuzzy logic and Fuzzy inference system	MMRE and BMMRE	Twenty software projects	Explored the use of fuzzy technique for software fault prediction. The presented system predicts faults in different phases of SDLC	The rules designed for fuzzy technique involved domain experts. However, no information about the domain experts has provided in the paper
Rathore and Kumar (2016b)	Decision tree regression	AAE, ARE, Kolmogorov–Smirnov test	18 datasets from PROMISE repository	Results showed that decision tree regression produced significant prediction accuracy for predictin of number of faults prediction in both intra-release and inter-release prediction	Only one technique has been investigated in the study. Comparative analysis is also limited

Table 5 Works related to severity of faults prediction

Study	Evaluation methods	Performance evaluation parameters	Dataset used	Advantages	Disadvantages
Szabo and Khoshgoftaar (1995)	Discriminant analysis	–	A data communications system	Classified software modules into different severity groups. Found that OO metrics enhanced the accuracy and reduce the misclassification rate	Criteria to classify software modules into different categories is not properly defined
Zhou and Leung (2006)	Logistic regression, Naive Bayes, random forest, and nearest neighbor with generalization	Precision, correctness, and completeness	KC1 form NASA data repository	Analyze OO metrics for predicting high and low severity faults. Found that CBO, WMC, RFC, and LCOM metrics are statistically significant for fault prediction across different severity levels	Empirical study with datasets of different domains is required to generalize the findings of the study
Shatnawi and Li (2008)	Logistic regression, multinomial regression, and Spearman correlations	ROC, sensitivity, specificity and regression coefficient	Eclipse project: Versions 2.0, 2.1, and 3.0	Results suggested that as a system evolves, the use of some commonly used metrics to identify which classes are more prone to errors becomes increasingly difficult	Fault data has been collected using some commercial tool. Thus, its accuracy is uncertain. Categorization of software modules into different severity categories is subjective

Table 5 continued

Study	Evaluation methods	Performance evaluation parameters	Dataset used	Advantages	Disadvantages
Jianhong et al. (2010)	5 different neural network based techniques	Mean absolute and root mean square errors, and accuracy	PC4 from NASA data repository	Explored the capabilities of neural network for severity of faults prediction. Results showed that resilient back propagation based technique produced best results	No concrete results presented. Evaluation of results is not thorough
Ardil et al. (2010)	Hybrid fuzzy-GA and neural network	Accuracy, MAE and RMSE	A dataset from NASA data repository	Presented an approach for the severity of fault prediction, which is rarely explored earlier	No detail about the fault dataset is provided. Results are evaluated only for one software project
Lamkanfi et al. (2011)	Naive Bayes, Naive Bayes multinomial, K-nearest neighbor, and support vector machines	Precision and recall	Mozilla, Eclipse, and GNOME	Compared different text mining techniques for predicting severity of bug. Results showed that overall, Naive Bayes produced the best fault prediction results	Software modules have been categorized into two severity categories only. Empirical study is not exhaustive
Sandhu et al. (2011)	Density-based spatial clustering with noise and neuro-fuzzy	accuracy	KC3 form NASA data repository	Both used techniques produced similar results for severity of faults prediction	Evaluation of the proposed techniques is limited
Gupta and Kang (2011)	A hybrid fuzzy-genetic algorithm and fuzzy clustering algorithm	Mean absolute error and root mean squared error	PC4 form NASA data repository	Results showed that fuzzy clustering technique produced best results as compared to other used techniques	Only one dataset has been used in the study. No comparative analysis presented

Table 5 continued

Study	Evaluation methods	Performance evaluation parameters	Dataset used	Advantages	Disadvantages
Wu (2011)	Decision tree (C5.0), Logistic regression,	Goodness-of-fit and Regression coefficient	KC1 form NASA data repository	Evaluated OO metrics for fault prediction. Found that WMC, NOC, LCOM, and CBO metrics were significant for fault prediction across all fault severity	The empirical study is limited to only one dataset
Yang et al. (2012)	Naive Bayes and three feature selection schemes, information gain, Chi-Square, and correlation coefficient	ROC, TPR, and FPR	Eclipse and Mozilla datasets	Results showed that used feature selection schemes improved the predication performance for over half the cases	The used feature selection schemes did not consider semantic relation during data extraction. Empirical study is not exhaustive
Chaturvedi and Singh (2012)	Naive Bayes, k-Nearest Neighbor, Naive Bayes Multinomial, SVM, J48, and RIPPER	Accuracy, precision, recall and F-measure	PITS datasets from NASA data repository	Concluded that the performance of machine learning techniques stabilized as we increase the number of terms 125 onwards	No significant improvements in the results reported
Shatnawi (2014)	Bayesian network, SVM, neural network, KNN, decision tree, CART, and random forest	ROC, Wilcoxon Rank Test	4 Ant, 3 Camel, and 1 Jedit projects	Proposed a method to classify software modules into different severity categories. Results showed that multi-categories prediction outperformed binary class prediction	Classification of software modules into different severity categories is subjective. Empirical study is limited and findings of are difficult to generalize

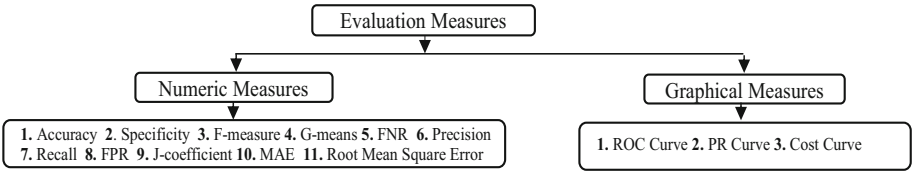

Fig. 5 Taxonomy of performance evaluation measures

cision, recall, j-coefficient, mean absolute error, and root mean square error. Graphical performance evaluation measures mainly include ROC curve, precision-recall curve, and cost curve.

5.1 Numeric measures

Numerical measures are the most commonly used measures to evaluate and validate fault prediction models (Jiang et al. 2008). The detail of these measures is given below.

Accuracy

Accuracy measures the probability of correctly classified fault-prone modules. But, it does not tell anything about incorrectly classified fault free modules (Olson 2008).

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{1}$$

However, the accuracy measure is somewhat ambiguous. For ex., if a classifier has achieved an accuracy of 0.8, then it means that 80% of the modules are correctly classified, while the status of the remaining 20% modules are remain unknown. Thus, if we are interested in misclassification cost also then accuracy is not a suitable measure (Jiang et al. 2008).

Mean absolute error (MAE) and root mean square error (RMSE)

The MAE measures the average magnitude of the errors in a set of prediction without considering their direction. It measures accuracy for continuous variables. The RMSE measures the average magnitude of the error. The difference between the predicted value and the actual value are squared and then averaged over the sample. Finally, the square root of the average is taken. Usually, MAE and RMSE measures are used together to provide a better picture of the error rates in fault prediction process.

Specificity, recall and precision

Specificity measures the fraction of correctly predicted fault-free modules. While sensitivity also known as recall or probability of detection (PD), measures probability that a module contained fault is classified correctly (Menzies et al. 2003).

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Specificity = \frac{TN}{FP + TN} \tag{3}$$

Precision measures the ratio of correctly classified faulty modules out of the modules classified as fault-prone.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Recall and specificity measures show the relation between type I and type II errors. It is possible to increases recall value by lowering precision and vise-versa (Menzies et al. 2007). Each of these three measures has independent consideration. However, the actual significance of these measures occurs when they use in combination.

G-mean, F-measure, H-measure and J-coefficient

To evaluate the fault prediction model for the imbalance datasets, G-means and F-measure, J-coefficient have been used. "G-mean1 is the square root of the product of recall and precision". "G-mean2 is the square root of the product of recall and specificity" (Kubat et al. 1998).

$$G\text{–}mean1 = \sqrt{(Recall * Precision)} \tag{5}$$

$$G\text{–}mean2 = \sqrt{(Recall * Specificity)} \tag{6}$$

F-measure provides the trade-off between classifier performances. It calculates the harmonic mean of precision and recall (Lewis and Gale 1994).

$$F\text{–}measure = \beta * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

In fault prediction, sometime a situation may occur that a classifier is achieving higher accuracy by predicting major class (non-faulty) correctly, while missing out the minor class (faulty). In this case, G-means and F-measure provide more honest scenario of prediction.

J-coefficient combines the performance index of recall and specificity (Youden 1950). It is defined by equation 8.

$$J\text{–}coefficient = Recall + Specificity - 1 \tag{8}$$

The value of J-coefficient $= 0$ implies that the probability of predicting a module faulty is equal to the false alarm rate. Whereas, the value of J-coefficient > 0 implies that classifier is useful for predicting faults.

FPR and FNR

The false positive rate is the faction of fault free modules that are predicted as faulty. The false negative rate is the ratio of module being faulty but predicted as non-faulty to total number of modules that are predicted as faulty.

$$FPR = \frac{FP}{FP + TN} \tag{9}$$

$$FNR = \frac{FN}{TP + FN} \tag{10}$$

5.2 Graphical measures

Graphical measures incorporated the techniques that show the visual trade-off of the correctly predicted fault-prone modules to the incorrectly predicted fault-free modules.

ROC curve and PR curve

Receiver Operator Characteristic (ROC) curve visualizes a trade-off between the number of correctly predicted faulty modules to the number of incorrectly predicted non-faulty modules (Yousef et al. 2004). In ROC curve, x-axis contained the False Positive Rate (FPR), while the y-axis contained the True Positive Rate (TPR). It provides an idea of overall model's performance by considering misclassification cost, if there is an unbalance in the

class distribution. The entire region of the curve is not important in software fault prediction point-of-view. Only, the area under the curve (AUC) within the region is used to evaluate the classifier performance.

In PR space, the recall is plots on the x-axis and the precision is on the y-axis. PR curve provides a more honest picture when dealing with high skewed data (Bockhorst and Craven 2005; Bunescu et al. 2005).

Cost curve

The numeric measures as well as ROC curve and PR curve ignored the impact of misclassification of faults on the software development cost. Certifying considerable number of faulty modules to be non-faulty raises serious concerns as it may result in the increment of a development cost due to the increase in fault removal cost of the same in the later phases. Jiang et al. (2008) have used various metrics to measure the performance of fault-prediction techniques. Later, they introduced cost curve to estimate the cost effectiveness of a classification technique. Drummond and Holte (2006) also proposed cost curve to visualize classifier performance and the cost of misclassification. Cost curve plots the probability cost function on the x-axis and the normalized expected misclassification cost on the y-axis.

Observations on performance evaluation measures

There are many evaluation parameters available that can be used to evaluate the prediction model performance. But, the selection of the best one is not a trivial task. Many factors influence the selection process such as how the class data is distributed, how the model built, how the model will use, etc. The comparison of different fault prediction model performance for predicting fault-prone software modules is the least studied areas in the software fault prediction literature (Arisholm et al. 2010b). Some of the works are available in the literature related to the analysis of the performance evaluation measures. Jiang et al. (2008) compared various performance evaluation measures for software fault prediction. Study found that no single performance evaluation measure able to evaluate the performance of fault prediction model completely. Combination of different performance measures can be used to calculate overall performance of fault prediction models. It is further added that rather than measuring model classification performance, we should focus on minimizing the misclassification cost and maximizing the effectiveness of software quality assurance process. In another study, Arisholm et al. (2010a) investigated various performance evaluation measures for software fault prediction. The results suggested that selection of best fault prediction technique or set of software metrics is highly dependent on the used performance evaluation measures. The selection of common evaluation parameters is still a critical issue in the context of software engineering experiments. The study suggested the use of performance evaluation measures that are closely linked to the intended, practical application of fault prediction model. Lessmann et al. (2008) also presented a study to evaluate performance measures for software fault prediction. They found that relying on accuracy indicators to evaluate fault prediction model is not appropriate. AUC was recommended as a primary indicator for comparing studies in software fault prediction.

Review of the studies related to performance evaluation measures suggests that we need more studies evaluating different performance measures in the context of software fault prediction. Since, the problems in the software fault prediction domain have inherited different issues compare to other's datasets, such as imbalance dataset, noisy data, etc. In addition, we need to focus more on the evaluation measures that incorporate misclassification rate and cost of erroneous prediction.

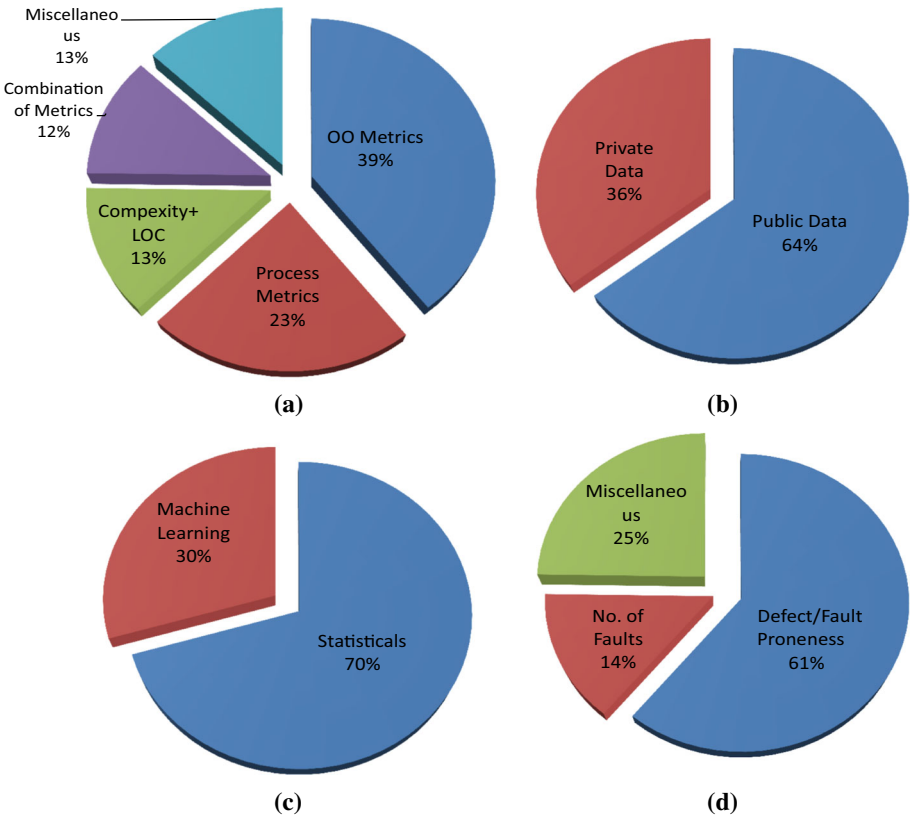

Fig. 6 Observations drawn from the software metrics studies

6 Statistical survey and observations

In the previous sections, we have presented an extensive study related to various dimensions of software fault prediction. The study and analysis have been performed with respect to software metrics (Sect. 3.1), data qualities issues (Sect. 3.3.2), software fault prediction techniques (Sect. 4), and performance evaluation measures (Sect. 5). Corresponding to each of the study, the observations and research gaps identified by us are also reported. Overall observations drawn from the statistical analysis of all these studies are shown in Figs. 6, 7 and 8.

Various observations drawn from the works reviewed for software metrics are shown in Fig. 6.

– As shown in Fig. 6a, object-oriented (OO) metrics are most widely studied and validated (39%) by the researchers, followed by process metrics (23%). The complexity and LOC metrics are the third largest set of metrics investigated by the researchers. While, combinations of different metrics (12%) least investigated by the researchers. One possible reason behind the high use of OO metrics for fault prediction is that traditional metrics (like Static code metrics, LOC metrics, etc.) did not capture the OO features such as inheritance, coupling, and cohesion, which are the root of the modern software develop-

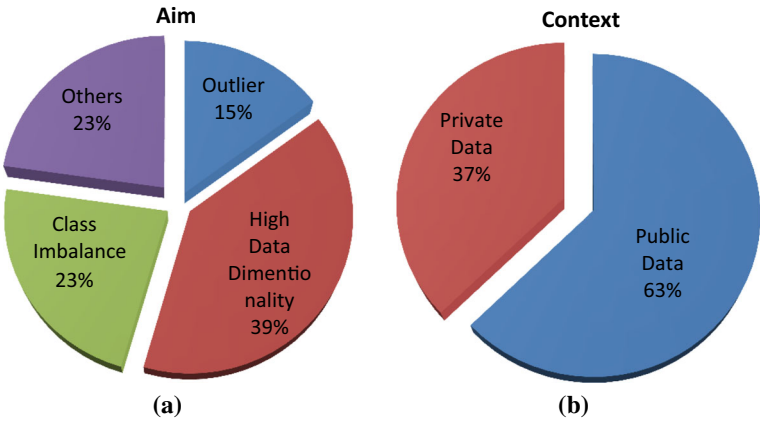

Fig. 7 Research focus of the studies related to data quality

ment practices. Whereas, OO metrics provide the measure of these features that help in the efficient OO software development process (Yasser A Khan and El-Attar 2011).

– As Fig. 6b shows that 64% of the studies used public datasets and only 36% used private datasets. While, a combination of the both types of datasets are the least used by studies (8%). Since, public datasets provide the benefit of replication of the studies and are easily available. It attracts the huge number of researchers to use the public datasets to perform their studies.
– It is revealed from Fig. 6c that the highest numbers of studies used statistical methods (70%) to evaluate and validate software metrics. While, only 30% of the studies used machine-learning techniques.
– It is clear from the Fig. 6d that capability of software metrics in predicting fault proneness has investigated by the highest number of researchers (61%). Only 14% of the studies investigated in the context of the number of fault prediction. While, 25% of the studies investigated other aspects of software fault prediction.

Various observations drawn from the works reviewed for data quality issues are shown in Fig. 7.

– As shown in Fig 7a, high data dimensionality is the primary data quality issue investigated by the researchers (39%). Class imbalance problem (23%) and outlier analysis (15%) are the second and third highly investigated data quality issues.
– As Fig. 7b shows that 63% of the studies investigated data quality issues have used public datasets and only 37% of the studies have used private or commercial datasets.

Various observations drawn from the works reviewed for software fault prediction techniques are shown in Fig. 8.

– It is shown in Fig. 8a that for performance evaluation measures, accuracy, precision and recall (46%) are the highest used by the researchers. AUC (15%) is the second highest used performance evaluation measure, while cost estimation (3%) and G-means (12%) are the least used by the researchers. The earlier researchers (before 2006–2007) have evaluated fault prediction models using simple measures such as accuracy, precision, etc., while, recent paradigm has been shifted to the use of performance evaluation measures such as cost curve, f-measure, AUC, etc. to evaluate fault prediction models (Jiang et al. 2008; Arisholm et al. 2010a).

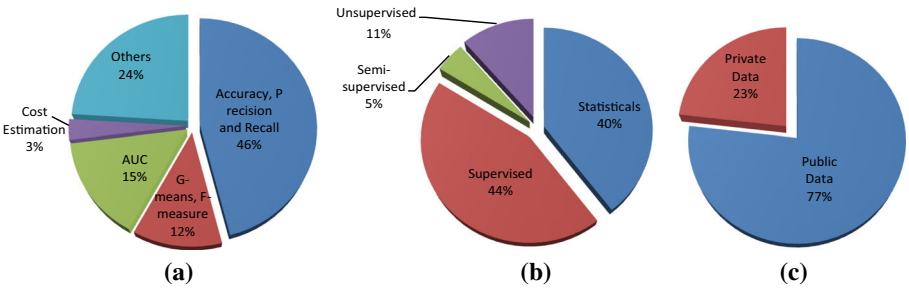

Fig. 8 Observations drawn from the studies related to software fault prediction techniques

- It is clear from Fig. 8b that supervised learning methods are the highest used by the earlier studies (44%) for building fault prediction models followed by statistical methods (40%). The reason behind the high use of statistical methods and supervised learning techniques for building fault prediction model is that they are simple to use and did not involve any complex parameter optimization. While, techniques like SVM, Clustering, etc. require a level of expertise before using them for fault prediction (Malhotra and Jain 2012; Tosun et al. 2010).
- Figure 8b shows that semi-supervised (5%) and unsupervised (11%) techniques have been used by fewer numbers of researchers. Since, the typical fault dataset contained the software metrics (independent variables) and the fault information (dependent variable). This makes it easy and suitable to use the supervised techniques for fault prediction. However, the use of semi-supervised and unsupervised techniques for fault prediction has increased recently.
- Figure 8c revels that 77% of the researchers have used public datasets to build fault prediction models and only 23% have used private datasets.

7 Discussion

Software fault prediction helps in reducing fault finding efforts by predicting faults prior to the testing process. It also helps in better utilization of testing resources and helps in streamlining software quality assurance (SQA) efforts to be applied in the later phases of software development. This practical significance of software fault prediction attracted a huge amount of research in this area in last two decades. The availability of open-source data repositories like NASA and PROMISE has also encouraged the researchers to perform studies and to draw out general conclusions. However, a large part of the earlier reported studies provided insufficient methodological details and thus made the task of software fault prediction difficult. The objective of this review work is to find the various dimensions of software fault prediction process and to analyze the works done in each of the dimension. We have performed an extensive search in various digital libraries to find out the studies published since 1993 and categorized them according to the targeted research areas. We excluded the papers from our study, which are not having the complete methodological detail or are not having experimental results. This leads us to narrow down our focus on the relevant papers only.

We observed that definition of software fault proneness is highly complex and ambiguous, and can be measured in different ways. A fault can be identified in any phase of software development. Some faults remain undetected during the testing phase and forwarded to the

field. One needs to understand the difference between pre-release and post-release faults before doing fault prediction. Moreover, the earlier prediction of faults is based on the binary class classification. This type of prediction provides an ambiguous picture of prediction, since, some modules are more fault-prone and require extra attention compared to the others. The more practical approach to the prediction should be based on the classification of the software modules based on the severity level of faults and prediction of the number of the faults. It can help to narrow down the SQA efforts to more severe modules and can result in a robust software system.

We have analyzed various studies reported for software fault prediction and found that the methodology used for software fault prediction affects the classifiers performance. There are three main concerns need attention before building any fault prediction model. First is the availability of a right set of datasets (detailed observations are given in Sect. 3.3.2 for data quality issues). It was found that fault datasets have lots of inheriting quality issues that lead to the poor prediction results. One needs to apply proper data cleaning and data preprocessing techniques to transform the data into the application context. In addition, fault prediction techniques needed to be selected according to the fault data in hand. Second is the optimal selection of independent variables (software metrics). A large number of software metrics are available. We can apply some feature selection techniques to select a significant set of metrics (detailed observations are given in Subsection 3.2.1 for software metrics). Last, the selection of fault prediction techniques should be optimized. One needs to extract the dataset characteristics and then select the fault prediction techniques based on the properties of the fault dataset (detailed observations are given in Subsection 4 for fault prediction techniques). Many accomplishments have been made in the area of software fault prediction, as highlighted throughout the paper. However, one question still remains to be answered, "why there has not been no big improvements or changing subproblems in software fault prediction". As discussed in their work, Menzies et al. (2008) showed that techniques/approaches used for fault prediction hit the "performance ceiling". Simply using better techniques does not guarantee the improved performance. Still, a large part of the community focuses on proposing or exploring new techniques for fault prediction. The author suggested that leveraging training data with more information content can help in breaking this performance ceiling. Next, most of the researchers focused on finding the solutions that are useful in the global context. In their study, Menzies et al. (2011) suggested that researchers should firstly check the validity of their solutions in the local context of the software project. Further, authors concluded that rather than seeking general solutions that can be applied to many projects, one should focus on finding the solutions that are best for the groups of related projects. The problem with fault prediction research does not lie in the used approaches, but on context in which fault prediction model build, performance measures used for model evaluation, and lack of the awareness of handling data quality issues. However, systematic methodologies are followed for software fault prediction, but a researcher must select a particular fault prediction approach by analyzing the context of the project and evaluate the prediction approach in the practical settings (e.g., how much effort do defect prediction models reduce for code review?) instead of only improving precision and recall.

8 Challenges and future directions in software fault prediction

In this section, we present some challenges and future directions in the software fault prediction. We also discuss some of the works done earlier to undertake these challenges.

(A) Adoption of software fault prediction for the rapidly changing environment of software development like Agile based development In recent years, the use of agile based approaches such as extreme programming, scrum, etc. has increased in software development and it has widely replaced the traditional software development practices. In conventional fault prediction process, historic data collected from the previous releases of the software project is used to build the model and to predict the faults in the current release of the software project. However, in agile based development, we follow very fast release cycle and hence sometimes enough data is not available for the early releases of the software project to build the fault prediction model. Therefore, it is needed to develop the methodologies that can predict faults in the early stages of software development. To solve this problem, Erturk and Sezer (2016) presented an approach that uses the expert knowledge and fuzzy inference system to predict faults in the early releases of software development and uses the conventional fault prediction process once the sufficient historic data is available. More such studies are needed to adopt the software fault prediction for agile-based development.

(B) Building fault prediction model to handle evolution of code bases One of the concerns with software fault prediction is the evolution of code bases. Suppose, we built a fault prediction model using a set of metrics and used it to predict the faults in given software project. However, some of these faults are fixed afterwards. Now, software system has evolved to accommodate the changes, but there may be the case when the values of the used set of metrics did not change. In that case, if we reuse the built fault prediction model, it will re-raise the same code area as fault-prone. This is a general problem of fault prediction models if we use the code metrics. Many of the studies presented in Table 1 (Sect. 3.2) used code metrics to build the fault prediction models. To solve this problem, it is needed to select the software metrics based on the development process and self-adapting measurements that capture already fixed faults. Recently, some researchers such as Nachiappan et al. (2010) and Matsumoto et al. (2010) proposed different sets of metrics such as software change metrics, file status metrics, developer metrics, etc. to build the fault prediction models. The future studies need to be done by using such software metrics to build the fault prediction models that capture the difference between two versions of the software project.

(C) Making fault prediction models more informative As Sect. 4 shows that many researchers have built fault prediction models for predicting software modules begin faulty and non-faulty. Only a few researchers focused on predicting number of faults and severity of faults. From the software practitioners perspective, sometimes it is beneficiary to know the modules having a large number of faults rather than simply having faulty or non-faulty information. It is often valuable for the software practitioner to know the modules having the largest number of faults since it would allow her/him to better identify most of the faults early and quickly. To solve this challenge, we need to make fault prediction models that are providing more information about the faultiness of software modules such as number of faults in a module, ranking of modules fault-wise, severity of a fault, etc. Some researchers such as Yang et al. (2015), Yan et al. (2010), Rathore and Kumar (2016b) presented their studies focusing on predicting ranking of software modules and number of defects prediction. Some studies related to fault prediction showed that a few number of modules contains most of the faults in software projects. Ostrand et al. (2005) presented a study to detect number of faults in top 20% of the files. We need more such type of studies to make the fault prediction models more informative.

(D) Considering new approaches to build fault prediction models As reported in Sect. 4, majority of fault prediction studies have used different machine learning and statistical techniques to perform the prediction. Recent results indicated that this current research paradigm, which relied on the use of straightforward machine learning techniques, has reached its

limit (Menzies et al. 2008). In the last decade, use of ensemble methods and multiple classifier combination approaches for fault prediction has gained considerable attention (Rathore and Kumar 2017). The studies related to these approaches reported better prediction performance compared to the individual techniques. In their study, Menzies et al. (2011) also suggested the use of additional information when building fault prediction models to achieve better prediction performance. However, there remains much work to do in this area to improve the performance of fault prediction models.

(E) Cross-company versus with-in company prediction The earlier fault prediction studies generally focused on the use of historical fault data of software project to build the prediction model. To employ this approach, a company should have maintained a data repository, where information about the software metrics and faults from the past projects or releases are stored. However, this practice is rarely followed by the software companies. In this situation, fault data from the different software project or different company can be used to build the fault prediction model for the given software project. The earlier reported studies related to this area found that the fault predictors learned from the cross-company data are not performing up to the mark. On the positive side, many researchers such as Zimmermann et al. (2009), Peters et al. (2013) have reported some studies to improve the performance of cross-company prediction. There remains much work to do in this area. We still need to handle issues such as selection of suitable training data for the projects without historic data, why fault prediction is not transitive, how to handle the data transfer of different project, etc. to make the cross-company prediction more effective.

(F) Use of search based approaches for fault prediction Search based approaches include the techniques from metaheuristic search, operations research and evolutionary computation to solve software fault prediction problem (Afzal 2011). These techniques model a problem in terms of an evaluation function and then use a search technique to minimize or maximize that function. Recently, some researchers have reported their studies using search based approaches for fault prediction (Afzal et al. 2010; Xiao and Afzal 2010). The results showed that an improved performance can be achieved using search based approaches. However, search based approaches typically require large number of evaluations to reach the solution. In Chen et al. (2017) presented a study to reduce the number of evaluations and to optimize the performance of these techniques. In future, researchers need to perform more studies using these approaches to examine what is the best way to use these approaches optimally. Such research can have a significant impact on the fault prediction performance.

9 Conclusions

The paper reviewed works related to various activities of software fault prediction such as software metrics, fault prediction techniques, data quality issues, and performance evaluation measures. We have highlighted various challenges and methodological issues associated with these activities of software fault prediction. From the survey and observations, it is revealed that most of the works have concentrated on OO metrics and process metrics using public data. Further, statistical techniques have been mostly used and they have mainly worked on the binary class classification. High data dimensionality and class imbalance quality issues have been widely investigated and most of the works have used accuracy, precision, and recall to evaluate the fault prediction performance. This paper also identified some challenges that can be explore by researchers in the future to make the software fault prediction process more evolved. The studies, reviews, survey, and observations enumerated in this paper can be helpful to the naive as well as established researchers of the related

field. From this extensive review, it can be concluded that more studies proposing and validating new software metrics sets by exploring the developers properties, cache history and location of faults, and other software development process related properties are needed. The future studies can try to build fault prediction models for cross-project prediction that can be useful for the organizations with insufficient fault project histories. The results of reviews performed in this work revealed that the performance of the different fault prediction techniques vary with different datasets. Hence, the work can be done to build the ensemble models for software fault prediction to overcome the limitations of individual fault prediction techniques.

Acknowledgements Authors are thankful to the MHRD, Government of India Grant for providing assistantship during the period this work was carried out. We are thankful to the editor and the anonymous reviewers for their valuable comments that helped in improvement of the paper.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Adrion WR, Branstad MA, Cherniavsky JC (1982) Validation, verification, and testing of computer software. ACM Comput Surv (CSUR) 14(2):159–192

Afzal W (2011) Search-based prediction of software quality: evaluations and comparisons. PhD thesis, Blekinge Institute of Technology

Afzal W, Torkar R, Feldt R, Wikstrand G (2010) Search-based prediction of fault-slip-through in large software projects. In: 2010 second international symposium on search based software engineering (SSBSE). IEEE, pp 79–88

Agarwal C (2008) Outlier analysis. Technical report, IBM

Ahsan S, Wotawa F (2011) Fault prediction capability of program file's logical-coupling metrics. In: Software measurement, 2011 joint conference of the 21st international workshop on and 6th international conference on software process and product measurement (IWSM-MENSURA), pp 257–262

Al Dallal J (2013) Incorporating transitive relations in low-level design-based class cohesion measurement. Softw Pract Exp 43(6):685–704

Alan O, Catal C (2009) An outlier detection algorithm based on object-oriented metrics thresholds. In: 24th international symposium on computer and information sciences, ISCIS'09, pp 567–570

Ardil E et al (2010) A soft computing approach for modeling of severity of faults in software systems. Int J Phys Sci 5(2):74–85

Arisholm E (2004) Dynamic coupling measurement for object-oriented software. IEEE Trans Softw Eng 30(8):491–506

Arisholm E, Briand L, Johannessen EB (2010a) A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. J Syst Softw 1:2–17

Arisholm E, Briand LC, Johannessen EB (2010b) A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. J Syst Softw 83(1):2–17

Armah GK, Guangchun L, Qin K (2013) Multi level data pre processing for software defect prediction. In: Proceedings of the 6th international conference on information management, innovation management and industrial engineering. IEEE Computer Society, pp 170–175

Bansiya J, Davis C (2002) A hierarchical model for object-oriented design quality assessment. IEEE Trans Softw Eng 28(1):4–17

Bibi S, Tsoumakas G, Stamelos I, Vlahvas I (2006) Software defect prediction using regression via classification. In: IEEE international conference on computer systems and applications, pp 330–336

Binkley A, Schach S (1998) Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures. In: Proceedings of the 20th international conference on software engineering, pp 452–455

Bird C, Nagappan N, Gall H, Murphy B, Devanbu P (2009) Putting it all together: using socio-technical networks to predict failures. In: Proceedings of the 2009 20th international symposium on software reliability engineering, ISSRE '09. IEEE Computer Society, Washington, pp 109–119

Bishnu PS, Bhattacherjee V (2012) Software fault prediction using quad tree-based k-means clustering algorithm. IEEE Trans Knowl Data Eng 24(6):1146–1151
Bockhorst J, Craven M (2005) Markov networks for detecting overlapping elements in sequence data. In: Proceeding of the neural information processing systems, pp 193–200
Briand L, Devanbu P, Melo W (1997) An investigation into coupling measures for C++. In: Proceeding of 19th international conference on software engineering, pp 412–421
Briand L, John W, Wust KJ (1998) An unified framework for cohesion measurement in object-oriented systems. Empir Softw Eng J 3(1):65–117
Briand L, Wst J, Lounis H (2001) Replicated case studies for investigating quality factors in object-oriented designs. Empir Softw Eng Int J 1:11–58
Bundschuh M, Dekkers C (2008) The IT measurement compendium: estimating and benchmarking success with functional size measurement. Springer
Bunescu R, Ruifang G, Rohit JK, Marcotte EM, Mooney RJ, Ramani AK, Wong YW (2005) Comparative experiments on learning information extractors for proteins and their interactions. Artif Intell Med (special issue on Summarization and Information Extraction from Medical Documents) 2:139–155
Caglayan B, Misirli TA, Bener A, Miranskyy A (2015) Predicting defective modules in different test phases. Softw Qual J 23(2):205–227
Calikli G, Bener A (2013) An algorithmic approach to missing data problem in modeling human aspects in software development. In: Proceedings of the 9th international conference on predictive models in software engineering, PROMISE '13. ACM, New York, pp 1–10
Calikli G, Tosun A, Bener A, Celik M (2009) The effect of granularity level on software defect prediction. In: 24th international symposium on computer and information sciences, ISCIS'09, pp 531–536
Canfora G, Lucia AD, Penta MD, Oliveto R, Panichella A, Panichella S (2013) Multi-objective cross-project defect prediction. In: Proceedings of the 2013 IEEE sixth international conference on software testing, verification and validation, ICST '13. IEEE Computer Society, Washington, pp 252–261
Catal C (2011) Software fault prediction: a literature review and current trends. Expert Syst Appl J 38(4):4626–4636
Catal C, Diri B (2007) Software fault prediction with object-oriented metrics based artificial immune recognition system. In: Product-focused software process improvement, vol 4589 of lecture notes in computer science. Springer, Berlin, pp 300–314
Catal C, Diri B (2008) A fault prediction model with limited fault data to improve test process. In: Product-focused software process improvement, vol 5089. Springer, Berlin pp 244–257
Catal C, Sevim U, Diri B (2009) Software fault prediction of unlabeled program modules. In Proceedings of the world congress on engineering, vol 1, pp 1–3
Challagulla V, Bastani F, Yen I-L, Paul R (2005) Empirical assessment of machine learning based software defect prediction techniques. In: 10th IEEE international workshop on object-oriented real-time dependable systems, WORDS'05, pp 263–270
Chatterjee S, Nigam S, Singh J, Upadhyaya L (2012) Software fault prediction using nonlinear autoregressive with exogenous inputs (narx) network. Appl Intell 37(1):121–129
Chaturvedi K, Singh V (2012) Determining bug severity using machine learning techniques. In: CSI sixth international conference on software engineering (CONSEG'12), pp 1–6
Chen J, Nair V, Menzies T (2017) Beyond evolutionary algorithms for search-based software engineering. arXiv preprint arXiv:1701.07950
Chidamber S, Darcy D, Kemerer C (1998) Managerial use of metrics for object oriented software: an exploratory analysis. IEEE Trans Softw Eng 24(8):629–639
Chidamber S, Kemerer C (1994) A metrics suite for object-oriented design. IEEE Trans Softw Eng 20(6):476–493
Chowdhury I, Zulkernine M (2011) Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. J Syst Archit 57(3):294–313
Couto C, Pires P, Valente MT, Bigonha RS, Anquetil N (2014) Predicting software defects with causality tests. J Syst Softw 93:24–41
Cruz AE, Ochimizu K (2009) Towards logistic regression models for predicting fault-prone code across software projects. In: 3rd international symposium on empirical software engineering and measurement ESEM'09, pp 460–463
Gray D, D. B., Davey N, Sun Y, Christianson B (2000) The misuse of the nasa metrics data program data sets for automated software defect prediction. In: Proceedings of 15th annual conference on evaluation and assessment in software engineering (EASE 2011. IEEE), pp 71–81
Dallal JA, Briand LC (2010) An object-oriented high-level design-based class cohesion metric. Inf Softw Technol 52(12):1346–361

Dejaeger K, Verbraken T, Baesens B (2013) Toward comprehensible software fault prediction models using bayesian network classifiers. IEEE Trans Softw Eng 39(2):237–257
Devine T, Goseva-Popstajanova K, Krishnan S, Lutz R, Li J (2012) An empirical study of pre-release software faults in an industrial product line. In: 2012 IEEE fifth international conference on software testing, verification and validation (ICST), pp 181–190
Drummond C, Holte RC (2006) Cost curves: an improved method for visualizing classifier performance. In: Machine learning, pp 95–130
Elish K, Elish M (2008) Predicting defect-prone software modules using support vector machines. J Syst Softw 81(5):649–660
Elish MO, Yafei AHA, Mulhem MA (2011) Empirical comparison of three metrics suites for fault prediction in packages of object-oriented systems: A case study of eclipse. Adv Eng Softw 42(10):852–859
Emam K, Melo W (1999) The prediction of faulty classes using object-oriented design metrics. In: Technical report: NRC 43609. NRC
Erturk E, Sezer EA (2015) A comparison of some soft computing methods for software fault prediction. Expert Syst Appl 42(4):1872–1879
Erturk E, Sezer EA (2016) Iterative software fault prediction with a hybrid approach. Appl Soft Comput 49:1020–1033
Euyseok H (2012) Software fault-proneness prediction using random forest. Int J Smart Home 6(4):1–6
Ganesh JP, Dugan JB (2007) Empirical analysis of software fault content and fault proneness using bayesian methods. IEEE Trans Softw Eng 33(10):675–686
Gao K, Khoshgoftaar TM (2007) A comprehensive empirical study of count models for software fault prediction. IEEE Trans Softw Eng 50(2):223–237
Gao K, Khoshgoftaar TM, Seliya N (2012) Predicting high-risk program modules by selecting the right software measurements. Softw Qual J 20(1):3–42
Glasberg D, Emam KE, Melo W, Madhavji N (1999) Validating object-oriented design metrics on a commercial java application. National Research Council Canada, Institute for Information Technology, pp 99–106
Graves T, Karr A, Marron J, Siy H (2000) Predicting fault incidence using software change history. IEEE Trans Softw Eng 26(7):653–661
Gray D, Bowes D, Davey N, Sun Y, Christianson B (2011) The misuse of the nasa metrics data program data sets for automated software defect prediction. In: 15th annual conference on evaluation assessment in software engineering (EASE'11), pp 96–103
Guo L, Cukic B, Singh H (2003) Predicting fault prone modules by the dempster–shafer belief networks. In: Proceedings of 18th IEEE international conference on automated software engineering, pp 249–252
Gupta K, Kang S (2011) Fuzzy clustering based approach for prediction of level of severity of faults in software systems. Int J Comput Electr Eng 3(6):845
Gyimothy T, Ferenc R, Siket (2005) Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Trans Softw Eng 31(10):897–910
Hall T, Beecham S, Bowes D, Gray D, Counsell S (2012) A systematic review of fault prediction performance in software engineering. IEEE Trans Softw Eng 38(6):1276–1304
Halstead MH (1977) Elements of software science (operating and programming systems series). Elsevier Science Inc., New York
Harrison R, Counsel JS (1998) An evaluation of the mood set of object-oriented software metrics. IEEE Trans Softw Eng 24(6):491–496
Hassan AE (2009) Predicting faults using the complexity of code changes. In: Proceedings of the 31st international conference on software engineering. IEEE Computer Society, pp 78–88
Herbold S (2013) Training data selection for cross-project defect prediction. The 9th international conference on predictive models in software engineering (PROMISE '13)
Huihua L, Bojan C, Culp M (2011) An iterative semi-supervised approach to software fault prediction. In: Proceedings of the 7th international conference on predictive models in software engineering, PROMISE '11, pp 1–15
Ihara A, Kamei Y, Monden A, Ohira M, Keung JW, Ubayashi N, Matsumoto KI (2012) An investigation on software bug-fix prediction for open source software projects—a case study on the eclipse project. In: APSEC workshops. IEEE, pp 112–119
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. Inf Sci J 176(24):3711–3734
Jiang Y, Cukic B, Yan M (2008) Techniques for evaluating fault prediction models. Empir Softw Eng J 13(5):561–595
Jianhong Z, Sandhu P, Rani S (2010) A neural network based approach for modeling of severity of defects in function based software systems. In: International conference on electronics and information engineering (ICEIE'10), vol 2, pp V2–568–V2–575

Johnson AM Jr, Malek M (1988) Survey of software tools for evaluating reliability, availability, and serviceability. ACM Comput Surv (CSUR) 20(4):227–269

Jureczko M (2011) Significance of different software metrics in defect prediction. Softw Eng Int J 1(1):86–95

Kamei Y, Sato H, Monden A, Kawaguchi S, Uwano H, Nagura M, Matsumoto K-I, Ubayashi N (2011) An empirical study of fault prediction with code clone metrics. In: Software measurement, 2011 joint conference of the 21st international workshop on and 6th international conference on software process and product measurement (IWSM-MENSURA), pp 55–61

Kamei Y, Shihab E (2016) Defect prediction: accomplishments and future challenges. In: Proceeding of 23rd international conference on software analysis, evolution, and reengineering, vol 5, pp 33–45

Kanmani S, Uthariaraj V, Sankaranarayanan V, Thambidurai P (2007) Object-oriented software fault prediction using neural networks. J Inf Softw Technol 49(5):483–492

Kehan G, Khoshgoftaar TM, Wang H, Seliya N (2011) Choosing software metrics for defect prediction: an investigation on feature selection techniques. Softw Pract Exp 41(5):579–606

Khoshgoftaar T, Gao K, Seliya N (2010) Attribute selection and imbalanced data: problems in software defect prediction. In: 2010 22nd IEEE international conference on, tools with artificial intelligence (ICTAI), vol 1, pp 137–144

Kim S, Zhang H, Wu R, Gong L (2011) Dealing with noise in defect prediction. In: Proceedings of the 2011 IEEE and ACM international conference on software engineering, ICSE '11. ACM, USA

Kitchenham B (2010) What's up with software metrics? A preliminary mapping study. J Syst Softw 83(1):37–51

Koru AG, Hongfang L (2005) An investigation of the effect of module size on defect prediction using static measures. In: Proceedings of the 2005 workshop on predictor models in software engineering, PROMISE '05, pp 1–5

Kpodjedo S, Ricca F, Antoniol G, Galinier P (2009) Evolution and search based metrics to improve defects prediction. In: 2009 1st international symposium on, search based software engineering, pp 23–32

Krishnan S, Strasburg C, Lutz RR, Govseva-Popstojanova K (2011) Are change metrics good predictors for an evolving software product line? In: Proceedings of the 7th international conference on predictive models in software engineering, promise '11. ACM, New York, pp 1–10

Kubat M, Holte RC, Matwin S (1998) Machine learning for the detection of oil spills in satellite radar images. Mach Learn J 30(2–3):195–215

Lamkanfi A, Demeyer S, Soetens Q, Verdonck T (2011) Comparing mining algorithms for predicting the severity of a reported bug. In: 2011 15th European conference on software maintenance and reengineering (CSMR), pp 249–258

Lessmann S, Baesens B, Mues C, Pietsch S (2008) Benchmarking classification models for software defect prediction: a proposed framework and novel findings. IEEE Trans Softw Eng 34(4):485–496

Lewis D, Gale WA (1994) A sequential algorithm for training text classifiers. In: Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '94, New York, NY, USA. Springer, New York, pp 3–12

Li M, Zhang H, Wu R, Zhou Z (2012) Sample-based software defect prediction with active and semi-supervised learning. Autom Softw Eng 19(2):201–230

Li W, Henry S (1993) Object-oriented metrics that predict maintainability. J Syst Softw 23(2):111–122

Li W, Henry W (1996) A validation of object-oriented design metrics as quality indicators. IEEE Trans Softw Eng 22(10):751–761

Li Z, Reformat M (2007) A practical method for the software fault-prediction. In: IEEE international conference on information reuse and integration, IRI'07. IEEE Systems, Man, and Cybernetics Society, pp 659–666

Liguo Y (2012) Using negative binomial regression analysis to predict software faults: a study of apache ant. Inf Technol Comput Sci 4(8):63–70

Lorenz M, Kidd J (1994) Object-oriented software metrics. Prentice Hall, Englewood Cliffs

Lu H, Cukic B (2012) An adaptive approach with active learning in software fault prediction. In: PROMISE. ACM, pp 79–88

Lu H, Cukic B, Culp M (2012) Software defect prediction using semi-supervised learning with dimension reduction. In: 2011 26th IEEE and ACM international conference on automated software engineering (ASE 2011), pp. 314–317

Ma Y, Luo G, Zeng X, Chen A (2012) Transfer learning for cross-company software defect prediction. Inf Softw Technol J 54(3):248–256

Ma Y, Zhu S, Qin K, Luo G (2014) Combining the requirement information for software defect estimation in design time. Inf Process Lett 114(9):469–474

Madeyski L, Jureczko M (2015) Which process metrics can significantly improve defect prediction models? an empirical study. Softw Qual J 23(3):393–422

Malhotra R, Jain A (2012) Fault prediction using statistical and machine learning methods for improving software quality. J Inf Process Syst 8(2):241–262
Marchesi M (1998) OOA metrics for the unified modeling language. In: Proceeding of 2nd Euromicro conference on Softwar eMaintenance and reengineering, pp 67–73
Martin R (1995) OO design quality metrics—an analysis of dependencies. Road 2(3):151–170
Matsumoto S, Kamei Y, Monden A, Matsumoto K, Nakamura M (2010) An analysis of developer metrics for fault prediction. In: PROMISE, p 18
McCabe T J (1976) A complexity measure. IEEE Trans Softw Eng SE–2(4):308–320
Mendes-Moreira J, Soares C, Jorge AM, Sousa JFD (2012) Ensemble approaches for regression: a survey. ACM Comput Surv (CSUR) 45(1):10
Menzies T, Butcher A, Marcus A, Zimmermann T, Cok D (2011) Local vs. global models for effort estimation and defect prediction. In: Proceedings of the 2011 26th IEEE/ACM international conference on automated software engineering, ASE '11. IEEE Computer Society, Washington, pp 343–351
Menzies T, DiStefano J, Orrego A, Chapman R (2004) Assessing predictors of software defects. In: Proceedings of workshop predictive software models
Menzies T, Greenwald J, Frank A (2007) Data mining static code attributes to learn defect predictors. IEEE Trans Softw Eng 33(1):2–13
Menzies T, Milton Z, Burak T, Cukic B, Jiang Y, Bener et al (2010) Defect prediction from static code features: current results, limitations, new approaches. Autom Softw Eng 17(4):375–407
Menzies T, Stefano J, Ammar K, McGill K, Callis P, Davis J, Chapman R (2003) When can we test less? In: Proceedings of 9th international software metrics symposium, pp 98–110
Menzies T, Turhan B, Bener A, Gay G, Cukic B, Jiang Y (2008) Implications of ceiling effects in defect predictors. In: Proceedings of the 4th international workshop on predictor models in software engineering, PROMISE '08. ACM, New York, pp 47–54
Mitchell A, Power JF (2006) A study of the influence of coverage on the relationship between static and dynamic coupling metrics. Sci Comput Program 59(1–2):4–25
Mizuno O, Hata H (2010) An empirical comparison of fault-prone module detection approaches: complexity metrics and text feature metrics. In: 2013 IEEE 37th annual computer software and applications conference, pp 248–249
Moreno-Torres JG, Raeder T, Alaiz-Rodrguez R, Chawla NV, Herrera F (2012) A unifying view on dataset shift in classification. Pattern Recogn 45(1):521–530
Moser R, Pedrycz W, Succi G (2008) A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: ICSE '08. ACM/IEEE 30th international conference on software engineering, 2008, pp 181–190
Nachiappan N, Zeller A, Zimmermann T, Herzig K, Murphy B (2010) Change bursts as defect predictors. In: Proceedings of the 2010 IEEE 21st international symposium on software reliability engineering, ISSRE '10. IEEE Computer Society, pp 309–318
Nagappan N, Ball T (2005) Use of relative code churn measures to predict system defect density. In: Proceedings of the 27th international conference on software engineering, ICSE '05. ACM, New York, pp 284–292
Nagappan N, Ball T, Zeller A (2006) Mining metrics to predict component failures. In: Proceedings of the 28th international conference on software engineering, ICSE '06. ACM, New York, pp 452–461
Nguyen THD, Adams B, Hassan AE (2010) A case study of bias in bug-fix datasets. In: Proceedings of the 2010 17th working conference on reverse engineering, WCRE '10. IEEE Computer Society, Washington, pp 259–268
Nikora A P, Munson J C (2006) Building high-quality software fault predictors. Softw Pract Exp 36(9):949–969
Nugroho A, Chaudron MRV, Arisholm E (2010) Assessing uml design metrics for predicting fault-prone classes in a java system. In: 2010 7th IEEE working conference on mining software repositories (MSR), pp 21–30
Ohlsson N, Zhao M, Helander M (1998) Application of multivariate analysis for software fault prediction. Softw Qual J 7(1):51–66
Olague HM, Etzkorn H, Gholston L, Quattlebaum S (2007) Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Trans Softw Eng 6:402–419
Olson D (2008) Advanced data mining techniques. Springer, Berlin
Ostrand TJ, Weyuker EJ, Bell RM (2004) Where the bugs are. In: Proceedings of 2004 international symposium on software testing and analysis, pp 86–96
Ostrand TJ, Weyuker EJ, Bell RM (2005) Predicting the location and number of faults in large software systems. IEEE Trans Softw Eng 31(4):340–355

Ostrand TJ, Weyuker EJ, Bell RM (2006) Looking for bugs in all the right places. In: Proceedings of 2006 international symposium on software testing and analysis, Portland, pp 61–72
Ostrand TJ, Weyuker EJ, Bell RM (2010) Programmer-based fault prediction. In: Proceedings of the 6th international conference on predictive models in software engineering, PROMISE '10. ACM, New York, pp 19–29
Pandey AK, Goyal NK (2010) Predicting fault-prone software module using data mining technique and fuzzy logic. Int J Comput Commun Technol 2(3):56–63
Panichella A, Oliveto R, Lucia AD (2014) Cross-project defect prediction models: L'union fait la force. In: 2014 software evolution week—IEEE conference on software maintenance, reengineering and reverse engineering (CSMR-WCRE), pp 164–173
Park M, Hong E (2014) Software fault prediction model using clustering algorithms determining the number of clusters automatically. Int J Softw Eng Appl 8(7):199–204
Peng H, Li B, Liu X, Chen J, Ma Y (2015) An empirical study on software defect prediction with a simplified metric set. Inf Softw Technol 59:170–190
Peters F, Menzies T, Marcus A (2013) Better cross company defect prediction. In: 10th IEEE working conference on mining software repositories (MSR'13), pp 409–418
Premraj R, Herzig K (2011) Network versus code metrics to predict defects: a replication study. In: 2011 international symposium on empirical software engineering and measurement (ESEM), pp 215–224
Radjenovic D, Hericko M, Torkar R, Zivkovic A (2013) Software fault prediction metrics: a systematic literature review. Inf Softw Technol 55(8):1397–1418
Rahman F, Devanbu P (2013) How, and why, process metrics are better. In: Proceedings of the 2013 international conference on software engineering, ICSE '13. IEEE Press, Piscataway, pp 432–441
Ramler R, Himmelbauer J (2013) Noise in bug report data and the impact on defect prediction results. In: 2013 joint conference of the 23rd international workshop on software measurement and the 2013 eighth international conference on software process and product measurement (IWSM-MENSURA), pp 173–180
Rana Z, Shamail S, Awais M (2009) Ineffectiveness of use of software science metrics as predictors of defects in object oriented software. In: WRI world congress on software engineering WCSE '09, vol 4, pp 3–7
Rathore S, Gupta A (2012a) Investigating object-oriented design metrics to predict fault-proneness of software modules. In: 2012 CSI sixth international conference on software engineering (CONSEG), pp 1–10
Rathore S, Gupta A (2012b) Validating the effectiveness of object-oriented metrics over multiple releases for predicting fault proneness. In: 2012 19th Asia-Pacific software engineering conference (APSEC), vol 1, pp 350–355
Rathore SS, Kumar S (2015a) Comparative analysis of neural network and genetic programming for number of software faults prediction. In: Recent advances in electronics & computer engineering (RAECE), 2015 national conference on. IEEE, pp 328–332
Rathore SS, Kumar S (2015b) Predicting number of faults in software system using genetic programming. Proced Comput Sci 62:303–311
Rathore SS, Kumar S (2016a) A decision tree logic based recommendation system to select software fault prediction techniques. Computing 99(3):1–31
Rathore SS, Kumar S (2016b) A decision tree regression based approach for the number of software faults prediction. SIGSOFT Softw Eng Notes 41(1):1–6
Rathore SS, Kumar S (2016c) An empirical study of some software fault prediction techniques for the number of faults prediction. Soft Comput 1–18. doi:10.1007/s00500-016-2284-x
Rathore SS, Kumar S (2017) Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems. Knowl Based Syst 119:232–256
Rodriguez D, Herraiz I, Harrison R (2012) On software engineering repositories and their open problems. In: 2012 first international workshop on realizing artificial intelligence synergies in software engineering, pp 52–56
Rodriguez D, Ruiz R, Cuadrado-Gallego J, Aguilar-Ruiz J, Garre M (2007) Attribute selection in software engineering datasets for detecting fault modules. In: Proceedings of the 33rd EUROMICRO conference on software engineering and advanced applications, EUROMICRO '07, pp 418–423
Rosenberg J (1997) Some misconceptions about lines of code. In: Proceedings of the 4th international symposium on software metrics, METRICS '97. IEEE Computer Society, Washington
Sandhu PS, Singh S, Budhija N (2011) Prediction of level of severity of faults in software systems using density based clustering. In: Proceedings of the 9th international conference on software and computer applications, IACSIT Press'11
Satria WR, Suryana HN (2014) Genetic feature selection for software defect prediction. Adv Sci Lett 20(1):239–244

Seiffert C, Khoshgoftaar T, Van Hulse J (2009) Improving software-quality predictions with data sampling and boosting. IEEE Trans Syst Man Cybern Part A Syst Hum 39(6):1283–1294

Seiffert C, Khoshgoftaar TM, Hulse JV, Napolitano A (2008) Building useful models from imbalanced data with sampling and boosting. In: Proceedings of the 21st international FLAIRS conference, FLAIRS'08. AAAI Organization

Seliya N, Khoshgoftaar TM (2007) Software quality estimation with limited fault data: a semi-supervised learning perspective. Softw Qual J 15:327–344

Selvarani R, Nair TRG, Prasad VK (2009) Estimation of defect proneness using design complexity measurements in object-oriented software. In: Proceedings of the 2009 international conference on signal processing systems, ICSPS '09. IEEE Computer Society, Washington, pp 766–770

Shanthi PM, Duraiswamy K (2011) An empirical validation of software quality metric suites on open source software for fault-proneness prediction in object oriented systems. Eur J Sci 51(2):168–181

Shatnawi R (2012) Improving software fault-prediction for imbalanced data. In: 2012 international conference on innovations in information technology (IIT), pp 54–59

Shatnawi R (2014) Empirical study of fault prediction for open-source systems using the chidamber and kemerer metrics. Softw IET 8(3):113–119

Shatnawi R, Li W (2008) The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process. J Syst Softw 11:1868–1882

Shatnawi R, Li W, Zhang H (2006) Predicting error probability in the eclipse project. In: Proceedings of the international conference on software engineering research and practice, pp 422–428

Shepperd M, Qinbao S, Zhongbin S, Mair C (2013) Data quality: some comments on the nasa software defect datasets. IEEE Trans Softw Eng 39(9):1208–1215

Shin Y, Bell R, Ostrand T, Weyuker E (2009) Does calling structure information improve the accuracy of fault prediction? In: 6th IEEE international working conference on mining software repositories, MSR '09, pp 61–70

Shin Y, Meneely A, Williams L, Osborne JA (2011) Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. IEEE Trans Softw Eng 37(6):772–787

Shin Y, Williams L (2013) Can traditional fault prediction models be used for vulnerability prediction? Empir Softw Eng J 18(1):25–59

Shivaji S, Jr, Akella JWE, R., Kim S (2009) Reducing features to improve bug prediction. In: Proceedings of the 2009 IEEE and ACM international conference on automated software engineering, ASE '09. IEEE Computer Society, Washington, pp 600–604

Singh P, Verma S (2012) Empirical investigation of fault prediction capability of object oriented metrics of open source software. In: 2012 international joint conference on computer science and software engineering, pp 323–327

Stuckman J, Wills K, Purtilo J (2013) Evaluating software product metrics with synthetic defect data. In: 2013 ACM and IEEE international symposium on empirical software engineering and measurement, vol 1

Sun Z, Song Q, Zhu X (2012) Using coding-based ensemble learning to improve software defect prediction. IEEE Trans Syst Man Cybern Part C Appl Rev 42(6):1806–1817

Swapna S, Gokhale, Michael RL (1997) Regression tree modeling for the prediction of software quality. In: Proceeding of ISSAT'97, pp 31–36

Szabo R, Khoshgoftaar T (1995) An assessment of software quality in a c++ environment. In: Proceedings sixth international symposium on software reliability engineering, pp 240–249

Tahir A, MacDonell SG (2012) A systematic mapping study on dynamic metrics and software quality. In: 28th IEEE international conference on software maintenance (ICSM), pp 326–335

Tang M, Kao MH, Chen MH (1999) An empirical study on object oriented metrics. In: Proceedings of the international symposium on software metrics, pp 242–249

Tang W, Khoshgoftaar TM (2004) Noise identification with the k-means algorithm. In: Proceedings of the 16th IEEE international conference on tools with artificial intelligence, ICTAI '04. IEEE Computer Society, Washington, pp 373–378

Tomaszewski P, Hakansson J, Lundberg L, Grahn H (2006) The accuracy of fault prediction in modified code—statistical model vs. expert estimation. In: 13th annual IEEE international symposium and workshop on engineering of computer based systems, 2006. ECBS 2006, pp 343–353

Tosun A, Bener A, Turhan B, Menzies T (2010) Practical considerations in deploying statistical methods for defect prediction: a case study within the turkish telecommunications industry. Inf Softw Technol 52(11):1242–1257 Special Section on Best Papers PROMISE 2009

Turhan B, Bener A (2009) Analysis of naive bayes' assumptions on software fault data: an empirical study. Data Knowl Eng 68(2):278–290

Vandecruys O, Martens D, Baesens B, Mues C, Backer M D, Haesen R (2008) Mining software repositories for comprehensible software fault prediction models. J Syst Softw 81(5):823–839 Software Process and Product Measurement

Venkata UB, Bastani BF, Yen IL (2006) A unified framework for defect data analysis using the mbr technique. In: Proceeding of 18th IEEE international conference on tools with artificial intelligence, ICTAI '06, 2006, pp 39–46

Verma R, Gupta A (2012) Software defect prediction using two level data pre-processing. In: 2012 international conference on recent advances in computing and software systems (RACSS), pp 311–317

Wang H, Khoshgoftaar T, Gao K (2010a) A comparative study of filter-based feature ranking techniques. In: 2010 IEEE international conference on information reuse and integration (IRI), pp 43–48

Wang H, Khoshgoftaar TM, Hulse JV (2010b) A comparative study of threshold-based feature selection techniques. In: Proceedings of the 2010 IEEE international conference on granular computing, GRC '10. IEEE Computer Society, Washington, pp 499–504

Wang S, Yao X (2013) Using class imbalance learning for software defect prediction. IEEE Trans Reliab 62(2):434–443

Wasikowski M, Chen X (2010) Combating the small sample class imbalance problem using feature selection. IEEE Trans Knowl Data Eng 22(10):1388–1400

Weyuker EJ, Ostrand TJ, Bell MR (2007) Using developer information as a factor for fault prediction. In: Proceedings of the third international workshop on predictor models in software engineering, PROMISE '07. IEEE Computer Society, Washington, pp 8–18

Wong W E, Horgan J R, Syring M, Zage W, Zage D (2000) Applying design metrics to predict fault-proneness: a case study on a large-scale software system. Softw Pract Exp 30(14):1587–1608

Wu F (2011) Empirical validation of object-oriented metrics on nasa for fault prediction. In:Tan H, Zhou M (eds) Advances in information technology and education, vol 201. Springer, Berlin, pp 168–175

Wu Y, Yang Y, Zhao Y, Lu H, Zhou Y, Xu B (2014) The influence of developer quality on software fault-proneness prediction. In: 2014 eighth international conference on software security and reliability (SERE), pp 11–19

Xia Y, Yan G, Jiang X, Yang Y (2014) A new metrics selection method for software defect prediction. In: 2014 International conference on progress in informatics and computing (PIC), pp 433–436

Xiao J, Afzal W (2010) Search-based resource scheduling for bug fixing tasks. In: 2010 second international symposium on search based software engineering (SSBSE). IEEE, pp 133–142

Xu Z, Khoshgoftaar TM, Allen EB (2000) Prediction of software faults using fuzzy nonlinear regression modeling. In: High assurance systems engineering, 2000, Fifth IEEE international symposium on. HASE 2000. IEEE, pp 281–290

Yacoub S, Ammar H, Robinson T (1999) Dynamic metrics for object-oriented designs. In: Proceeding of the 6th international symposium on software metrics (Metrics'99), pp 50–60

Yadav HB, Yadav DK (2015) A fuzzy logic based approach for phase-wise software defects prediction using software metrics. Inf Softw Technol 63:44–57

Yan M, Guo L, Cukic B (2007) Statistical framework for the prediction of fault-proneness. In: Advance in machine learning application in software engineering. Idea Group

Yan Z, Chen X, Guo P (2010) Software defect prediction using fuzzy support vector regression. In: International symposium on neural networks. Springer, pp 17–24

Yang C, Hou C, Kao W, Chen I (2012) An empirical study on improving severity prediction of defect reports using feature selection. In: 2012 19th Asia-Pacific software engineering conference (APSEC), vol 1, pp 350–355

Yang X, Tang K, Yao X (2015) A learning-to-rank approach to software defect prediction. IEEE Trans Reliab 64(1):234–246

Yasser A Khan, MOE, El-Attar M (2011) A systematic review on the relationships between ck metrics and external software quality attributes. Technical report

Youden WJ (1950) Index for rating diagnostic tests. Cancer 3(1):32–35

Yousef W, Wagner R, Loew M (2004) Comparison of non-parametric methods for assessing classifier performance in terms of roc parameters. In: Proceedings of international symposium on information theory, 2004. ISIT 2004, pp 190–195

Zhang H (2009) An investigation of the relationships between lines of code and defects. In: IEEE international conference on software maintenance (ICSM), pp 274–283

Zhang W, Yang Y, Wang Q (2011) Handling missing data in software effort prediction with naive Bayes and EM algorithm. In: Proceedings of the 7th international conference on predictive models in software engineering, PROMISE '11. ACM, New York, pp 1–10

Zhang X, Gupta N, Gupta R (2007) Locating faulty code by multiple points slicing. Softw Pract Exp 37(9):935–961

Zhimin H, Fengdi S, Yang Y, Li M, Wang Q (2012) An investigation on the feasibility of cross-project defect prediction. Autom Software Eng 19(2):167–199

Zhou Y, Leung H (2006) Empirical analysis of object-oriented design metrics for predicting high and low severity faults. IEEE Trans Softw Eng 10:771–789

Zhou Y, Xu B, Leung H (2010) On the ability of complexity metrics to predict fault-prone classes in object oriented systems. J Syst Softw 83(4):660–674

Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B (2009) Cross-project defect prediction: A large scale experiment on data vs. domain vs. process. In: Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, ESEC and FSE '09. ACM, New York, pp 91–100