A Critical Analysis on Software Fault Prediction Techniques

Pradeep Kumar Singh, Ranjan Kumar Panda and Om Prakash Sangwan

1Department of CSE, ASET, Amity University Uttar Pradesh, Noida, India
2Gautam Buddha University, Greater Noida, India

Abstract: This paper presents the results of a systematic review conducted to collect evidence on software fault prediction techniques. Different models, methods, algorithms and approaches were studied and conclusion was drawn. The review was conducted by studying the different set of parameters at class level, component level and other software fault prediction techniques considering object oriented design approach. The information was collected from various research papers related to fault prediction and out of 577 mainly 15 studies, which were found most relevant are analyzed. Our results shows that there are few metrics which helps in predicting early fault prediction in software and reduces testing cost, increases reliability, quality of software, helps development managers in decision making to identify and make strategic planning regarding most faulty modules.

Key words: Software faults • Fault predictions • Defects • Software maintainability • Cost • Software reliability • Software quality

INTRODUCTION

By definition, a fault is a structural imperfection in a software system that may lead to the systems eventually failing [22].

For a large software system it is important to predict faults well in advance before it is delivered to the users or customers. This will reduce the cost of test efforts as the defective modules are early detected [7]. During software development emphasis can be given to most faulty modules when these are early detected. Software quality increases due to the early detection of faults in defective modules before testing phase and hence reliability of software increases [9]. Software fault prediction helps in reducing the effort in maintainability and increases reliability.

Systematic Review (SR): A system review is used to identify, analyze the research those have been already done on fault prediction techniques for software and conclusion is drawn based on the research performed on the available facts on the conducted studies. This section mentions the detail steps followed to perform the systematic review.

General Research Questions (RQ) on Software Faults:
Before starting a SR many research questions were framed related to the software fault prediction. The solutions to the questions may be addressed by our SR. The questions which are not addressed by SR are put forward for further research.

The following research questions (RQ) have been addressed in our studies.

RQ1: What are the techniques/methods those have been used for early fault predictions in software?
RQ2: What are the parameters or metrics those affect/related to fault prediction in software?
RQ3: What are the metrics to identify a superior method of fault prediction and decide its accuracy measures?
RQ4: How fault prediction helps in reduction of cost of test effort and positively affects timely delivery?
RQ5: How fault prediction helps in decision making and increases reliability and quality of software applications?

Search Strategies for Preliminary Studies: Before performing SR the below strategies were followed to collect information for preliminary studies. Relevant
research papers from various data sources were collected by the help of below searching strategies. The keywords related to fault predictions were used in most relevant papers. The synonyms for the keywords were considered in search operations.

Some keywords were used for giving emphasis on object oriented software approach. Few search strings like fault in softwares, defect prediction, software defects were used to extract the related research papers from various databases. In addition to this we have collected the various research papers by adopting the similar approach as adopted by Singh et al. [24, 25]

Information Collection: Information regarding software fault prediction were collected from two sources called primary source and secondary source.

Primary Source of Information: Research papers from IEEE digital Library, ACM digital Library, Springer link, Science Direct were searched on the basis of search strategies mentioned above. The most relevant papers related to software fault prediction techniques, its impact on software cost and quality were selected for SR review in this paper. Total 577 papers were extracted during search operation. Papers those did not meet the search criteria were excluded for review. Total 15 most relevant papers were selected for SR review in this paper.

Secondary Source of Information: The research papers selected for SR review contains many reference papers which are also related to software fault predictions and relevant to our SR review. Those papers were also referred for our SR reviews.

RESULTS

The summary of search result is shown in Table 1 which includes total papers searched, duplicate papers and relevant papers mentioning their corresponding database names. In our research work 15 papers were selected which contain the studies done so far from year 2002 to 2014 related to various methodology, techniques or models related to software fault prediction.

Inclusion and Exclusion Criteria for Study Selection: The criteria’s to include studies in our SR were to include any study that was either: 1. Software fault predictions techniques OR software defects prediction AND software quality OR 2. Software fault predictions AND cost AND software reliability and terms containing similar terminology. Any study those discussed other than search key words mentioned in this paper were excluded.

Study Selection: The study selection was performed in two phases: initial selection and final selection. In initial selection the title and abstract of the papers were thoroughly checked applying the search key words and search criteria’s as mentioned earlier. Total 15 paper including one technical papers were selected in primary search process. During the secondary search process the references and citation of these 15 papers were reviewed and another 4 papers were included.

Study Quality Assessment: Nineteen selected papers were assessed with quality assessment procedure. There were 11 criteria’s in the quality assessment procedure as mentioned by Dybå and Dingsøyr (Dybå and Dingsøyr, 2008) [23]. Due to space constraints quality study for individual papers for each question is not mentioned in this paper. Each question is assessed with a ‘Yes’ or ‘No’ answer and have a score value of ‘1’ or ‘0’ respectively. Therefore, the study could score between values 0 and 11. We have decided a minimum quality score as 3 as cut off criteria for paper selection for our SR. Out of total 20 papers 15 papers were selected and 5 papers were discarded which did not obtain minimum score of 3 in quality assessment study. Table 1.1 gives a summary of the scores obtained for each selected study.

<table>
<thead>
<tr>
<th>No</th>
<th>Source of databases</th>
<th>No. of search results retrieved</th>
<th>No. of duplicate s found</th>
<th>Number of relevant research papers found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IEEE Xplore</td>
<td>354</td>
<td>25</td>
<td>09</td>
</tr>
<tr>
<td>2</td>
<td>ACM digital Library</td>
<td>155</td>
<td>8</td>
<td>03</td>
</tr>
<tr>
<td>3</td>
<td>Springer link</td>
<td>25</td>
<td>4</td>
<td>01</td>
</tr>
<tr>
<td>4</td>
<td>Science Direct</td>
<td>43</td>
<td>7</td>
<td>02</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>577</td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Study Id</th>
<th>ID1</th>
<th>ID2</th>
<th>ID3</th>
<th>ID4</th>
<th>ID5</th>
<th>ID6</th>
<th>ID7</th>
<th>ID8</th>
<th>ID9</th>
<th>ID10</th>
<th>ID11</th>
<th>ID12</th>
<th>ID13</th>
<th>ID14</th>
<th>ID15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total score</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
Analysis Based on Research

Research Question 1: Research Question 1 of the SR was intended to find evidence on software fault prediction techniques/methods or models. The results of the data obtained are briefly mentioned in Table 2.

Statistical models were used in ID3, ID9. Regression tree method were used in ID1. Object oriented methods were used in ID5, ID10, ID13. Fuzzy inference method were used in ID7. Other fault detection and removal models were used in ID8, ID11, ID13, ID15. Fault prediction based on history were used in ID4.

After analyzing the findings of Table 2 and answering to Question 1 of SR review below facts can be mentioned briefly.

- Most commonly used design metrics at class level for object oriented (OO) software are CBO, NOC, WMC, RFC, DIT, LC, LOC. These metrics are often studied for fault prediction in object oriented software systems [3, 10, 13].

Research Question 2: Question 2 of the SR was intended to find evidence on software fault prediction parameters.

The different predictors of software fault at class level and component level are used in ID3. These are mentioned in Table 3.

Predictors based on calling structure, history and non-calling structure, human experts are used in ID4. Fault prediction in software systems also depends on faulty or skewed data which are used in software as studied in ID5 and ID9. Fault prediction depends on the tokens of source codes written for the software system mentioned in ID2. Detection and removal of leading faults and thus removing dependent faults as mentioned in ID8. Ranking of fault prone modules and thus identifying most faulty module helps in better elimination of faults as mentioned in ID7. Most commonly used product metrics for fault prediction is CK metrics as mentioned in ID13, ID14 and also depends on process metrics as mentioned in ID14.

<table>
<thead>
<tr>
<th>Serial Number</th>
<th>Study ID</th>
<th>Fault prediction Features</th>
<th>Methods/Models/Techniques Used</th>
<th>Authors</th>
<th>Source</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ID3[3]</td>
<td>Fault detection in software using statistical and evaluation methods</td>
<td>1. Statistical model- parameters considered for fault detection in class level like Coupling Number of Children (NOC) Weighted Methods per Class (WMC) Response for Class (RFC) Depth of Inheritance Tree (DIT) Lack of Cohesion (LC) Number of statements (LOC) Maximum cyclomatic complexity. 2. Evaluation by experts 3. Statistical model- parameters considered for fault detection in component level like Number of statements/Number of methods, Number of modified classes in the component</td>
<td>Tomaszewski et al.</td>
<td>IEEE Xplore</td>
<td>2006</td>
</tr>
<tr>
<td>4</td>
<td>ID4[4]</td>
<td>Fault prediction based on history and calling and non-calling structure parameters.</td>
<td>1. Based on calling structure parameters like Callers, Prior Changed Callers, Prior Faulty Callers. 2. Based on history like 2.1 Based on source code characteristics like KLOC, Release number, New file status in the prior release, Number of changes in the prior release, number of code changes, Cumulative number of developers changing file prior to the current release.</td>
<td>Shin et al.</td>
<td>IEEE Xplore</td>
<td>2009</td>
</tr>
<tr>
<td>5</td>
<td>ID9[9]</td>
<td>Fault prediction and hence software quality depends on fault proneness of data.</td>
<td>The statistical method, machine learning methods, neural network techniques and clustering techniques were applied on below two metrics with real time defect datasets of NASA software projects, JM1, PCI and CM1 1. Requirement matrix 2. Code matrix</td>
<td>Brar et al.</td>
<td>IEEE Xplore</td>
<td>2009</td>
</tr>
</tbody>
</table>
### Table 2: Continue

<table>
<thead>
<tr>
<th>ID</th>
<th>Defect prediction in high assurance software</th>
<th>Different defect prediction models used are</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1. Roughly balanced bagging (RBBag) algorithm.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2. C4.5 learner</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3. Naïve Bayes learner</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Sellya et al. IEEE Xplore 2010</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Ranking of fault prone modules</th>
<th>Prediction and ranking of fault prone modules are done using below methods.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1. ID3 algorithm combined with Fuzzy inference system</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Pandey et al. IEEE Xplore 2010</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Prediction of fault proneness for imbalance data</th>
<th>Applied on object oriented (OO) software systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1. Fault content method.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2. SMOTE (synthetic minority over-sampling technique)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Shattawli et al. IEEE Xplore 2012</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Fault Prediction Capabilities of Five Prediction Models for Software Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Five fault prediction models were used</td>
</tr>
<tr>
<td>10</td>
<td>1. Simple Logistic</td>
</tr>
<tr>
<td>11</td>
<td>2. K-means Clustering</td>
</tr>
<tr>
<td>12</td>
<td>3. C4.5 Decision Tree Algorithm</td>
</tr>
<tr>
<td>13</td>
<td>4. Random Forest</td>
</tr>
<tr>
<td>14</td>
<td>5. Neural Network</td>
</tr>
<tr>
<td>15</td>
<td>Banthia et al. ACM 2012</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>To detect fault in modules from tokens in source code.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Below two methods were used to detect faults on the basis of source code tokens.</td>
</tr>
<tr>
<td>11</td>
<td>1. Multi-variate Bernoulli model</td>
</tr>
<tr>
<td>12</td>
<td>2. Multinomial model.</td>
</tr>
<tr>
<td>13</td>
<td>Mirano et al. IEEE Xplore 2013</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature Ranking and Feature Subset Selection for Fault prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1. Seven different feature ranking techniques:</td>
</tr>
<tr>
<td>12</td>
<td>Chi-square (CS)</td>
</tr>
<tr>
<td>13</td>
<td>Info Gain (IG)</td>
</tr>
<tr>
<td>14</td>
<td>Gain Ratio (GR)</td>
</tr>
<tr>
<td>15</td>
<td>ReliefF (RF)</td>
</tr>
<tr>
<td>16</td>
<td>SVM Ouch and Principal Component Analysis (PCA)</td>
</tr>
<tr>
<td>17</td>
<td>2. Eighteen metrics of an object-oriented software System were used as below.</td>
</tr>
<tr>
<td>18</td>
<td>WMC, CBO, RFC, DIT, NOC, IC, CA, CE, MFA, LCOM1, LCOM2, LCOM3, CAMO3, NPM, DAM, AMC, LOC and maxCC.</td>
</tr>
<tr>
<td>19</td>
<td>3. Feature subset selection was done by Logistic Regression(LR) algorithm</td>
</tr>
<tr>
<td>20</td>
<td>Singh et al. ACM 2014</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Search Based Techniques(SBT) for Software Fault Prediction models</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Below Search Based Techniques were used for developing software fault prediction models.</td>
</tr>
<tr>
<td>13</td>
<td>1. Genetic Programming (GP)</td>
</tr>
<tr>
<td>14</td>
<td>2. Ant Colony Optimization (ACO).</td>
</tr>
<tr>
<td>15</td>
<td>3. Multi-Objective Particle Swarm Optimization (MOPSO)</td>
</tr>
<tr>
<td>16</td>
<td>4. Genetic Algorithms (GA)</td>
</tr>
<tr>
<td>18</td>
<td>6. Simulated Annealing- Probabilistic Neural Network (SA-PNN)</td>
</tr>
<tr>
<td>19</td>
<td>Malhotra et al. ACM 2014</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Comparison of Software Fault Prediction Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>For fault prediction K-Nearest Neighbor (KNN) and multiple linear regression (MLR) were applied on</td>
</tr>
<tr>
<td>14</td>
<td>1. Six CK metrics</td>
</tr>
<tr>
<td>15</td>
<td>2. Eleven Object oriented metrics.</td>
</tr>
<tr>
<td>16</td>
<td>Below Statistical measure were applied R2, Adjusted R2, root mean square (RMSE) and root relative squared error (RRSE).</td>
</tr>
<tr>
<td>17</td>
<td>Below four Java based software module were used:</td>
</tr>
<tr>
<td>19</td>
<td>Goyala et al. Science Direct 2014</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Fault prediction based on product and process metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Below product metrics were used.</td>
</tr>
<tr>
<td>15</td>
<td>1. CK metrics</td>
</tr>
<tr>
<td>16</td>
<td>2. Lack of Cohesion in Methods (LCOM3)</td>
</tr>
<tr>
<td>17</td>
<td>3. QMOOD</td>
</tr>
<tr>
<td>18</td>
<td>4. The quality oriented extension of CK metrics.</td>
</tr>
<tr>
<td>19</td>
<td>5. Coupling metrics.</td>
</tr>
<tr>
<td>20</td>
<td>6. Class level metrics based on complexity metric.</td>
</tr>
<tr>
<td>22</td>
<td>Below process metrics were used</td>
</tr>
<tr>
<td>23</td>
<td>1. Number of Revisions (NR).</td>
</tr>
<tr>
<td>24</td>
<td>2. Number of Distinct Committers (NDC).</td>
</tr>
<tr>
<td>25</td>
<td>3. Number of Modified Lines (NML).</td>
</tr>
<tr>
<td>26</td>
<td>4. Number of Defects in Previous Version (NDFV)</td>
</tr>
<tr>
<td>27</td>
<td>5. Madeyski et al. Springer 2014</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Fault detection process(FDP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>And Fault correction process(FCP) along with testing effort function and imperfect debugging</td>
</tr>
<tr>
<td>16</td>
<td>1. Below Testing effort functions(TEF) were studied</td>
</tr>
<tr>
<td>17</td>
<td>1.1. Constant TEF</td>
</tr>
<tr>
<td>18</td>
<td>1.2. Weibull TEF</td>
</tr>
<tr>
<td>19</td>
<td>1.3. Logistic TEF</td>
</tr>
<tr>
<td>20</td>
<td>2. Study was done between cumulative no. of detected faults versus Testing time(weeks) by computing FDP model and actual data.</td>
</tr>
<tr>
<td>21</td>
<td>3. Study was done between cumulative no. of detected faults versus Testing time(weeks) by comparing FCP model and actual data.</td>
</tr>
<tr>
<td>22</td>
<td>Peng et al. Science Direct 2014</td>
</tr>
</tbody>
</table>
Table 3: Summary of Evidence in support of Predictors at class level and component level

<table>
<thead>
<tr>
<th>Study ID</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID3</td>
<td>Predictors at the class level are CBO, NOC, WMC, RFC, DIT, LC, LOC and Maximum cyclomatic complexity.</td>
</tr>
<tr>
<td></td>
<td>Parameters at the component level like Number of statements, Number of methods, Number of modified classes in the component.</td>
</tr>
<tr>
<td>ID4</td>
<td>Based on calling structure parameters like: Callees, Callers, PriorChangedCallees, PriorChangedCallers, PriorFaultyCallees, PriorFaultyCallers.</td>
</tr>
<tr>
<td>ID4</td>
<td>Parameters based on non-calling structure like KLOC, Release number, New file status in the prior release, Number of changes in the prior release, Number of code changes, Cumulative number of developers changing file prior to the current release.</td>
</tr>
</tbody>
</table>

In summary the fault prediction in software system depends on below parameters or metrics and thus answer to Research Question 2 is addressed below. Predictors at class and component levels as mentioned in [3]. Predictors based on calling structure, history and non calling structure human experts as mentioned in [4]. Fault prediction in software systems also depends on faulty or skewed data which are used in software as studied in [5, 9]. Fault prediction depends on the tokens of source codes written for the software system as mentioned in [2]. Detection and removal of leading faults and thus removing dependent faults automatically and easily as mentioned in [8]. Ranking of fault prone modules and thus identifying most faulty module helps in better elimination of faults as mentioned in [7].

Most commonly used product metrics for fault prediction is CK metrics and also depends on process metrics as mentioned in [13, 14].

**Research Question 3:** Question 3 of the SR was intended to find evidence on how to identify a superior method in software fault prediction and decide the accuracy measures. The answer to Question 3 is addressed in Table 4.

The result of SR regarding superior models/methods and their accuracy measures have been discussed briefly in Table 4.

**Research Question 4:** Question 4 of the SR was intended to find evidence on how fault prediction helps in reduction of cost of test effort and positively affects timely delivery?

Summary of evidence in support of test effort reduction and on time delivery of software system and thus Question 4 is addressed below (studied in ID7 and ID8).

The most faulty modules are identified and ranked. Thus it helps the development manager to make strategies to eliminate defects in this modules. This saves the testing effort in later test phase and also saves the delivery time as mentioned in [7].

As per Pareto principle approximately 80% of the effects come from 20% of the causes. Identifying leading faults and removing them helps to eliminate dependent faults as well. Thus it saves testing cost and delivery time for the software project as mentioned in [8].

**Research Question 5:** Question 5 of the SR was intended to find evidence on how fault prediction helps in decision making and increases reliability and quality of software applications?

Summary of evidence in support of building fault prediction models which increases reliability and quality of software applications and thus Question 5 is addressed below.

SRGM (software reliability growth models) as mentioned in [8]. Prediction model which is combination of requirement metric and code metric applied with K means clustering [9].

Software fault prediction models built using search based techniques and Area under the Curve method (AUC) [12].

**Threats to Validity:** The main threat to the validity of this SR review is related to bias in selecting papers through inclusion and exclusion criteria. There might be slight risk
<table>
<thead>
<tr>
<th>Study ID</th>
<th>Methods/Models/Techniques Used</th>
<th>Accuracy measure</th>
<th>Value of Accuracy Measure</th>
<th>Model predicted to be superior</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID1</td>
<td>1. Regression tree method 2. cart(bite-length squares) 3. s-plus 4. Cart(lad)(least absolute deviation)</td>
<td>aae (p = 0.496)</td>
<td>Cart-lad tree model has better fault predictive accuracy.</td>
<td></td>
</tr>
<tr>
<td>ID2</td>
<td>For fault prediction below models were used. 1. Multi-variate Bernoulli model 2. Multinomial model.</td>
<td>Average PIV(TCALL value = 0.231, TCCODE = 0.268)</td>
<td>Multi-variate Bernoulli model is a better model for fault prediction.</td>
<td></td>
</tr>
<tr>
<td>ID3</td>
<td>1. Statistical model - parameters considered for fault detection in class level like Coupling, Number of Childen, Weighted Methods per Class, Response for Class, Depth of Inheritance Tree, Lack of Cohesion, Number of statements, Maximum cyclomatic complexity. 2. Evaluation by experts 3. Statistical model - parameters considered for fault detection in component level like Number of statements, Number of methods, Number of modified classes in the component</td>
<td>% of code to analyze versus % of faults found at component level for 60% of total code analysis.</td>
<td>Statistical model at the class level better predicts fault than at component level and evaluation by human expert.</td>
<td></td>
</tr>
<tr>
<td>ID4</td>
<td>1. Based on calling structure parameters like Callees, Callers, Prior Changed Callees, Prior Changed Callers, Prior Faulty Callees, Prior Faulty Callers. 2. Based on history like KLOC, Release number, New file status in the prior release, Number of changes in the prior release, number of code changes, Cumulative number of developers changing file prior to the current release.</td>
<td>History, calling structure and code attributes have higher rate of fault prediction than Calling structure &amp; code attributes. Prediction results with non-calling structure attributes (Total Improvement = 8.73, Δ(%) = 12%)</td>
<td>History model shows 4.5% better fault prediction accuracy than the methods of calling structures. History model combined with non-calling structure methods is better fault predictor than calling structure method.</td>
<td></td>
</tr>
<tr>
<td>ID5</td>
<td>1. Fault content method. 2. SMOTE (synthetic minority over-sampling technique)</td>
<td>Classifiers used Naive Bayes, Nearest Neighbor (1NN), Nearest Neighbor (5 NN)</td>
<td>Fault content method is better in predicting the most faulty module in an object oriented software.</td>
<td></td>
</tr>
<tr>
<td>ID6</td>
<td>1. Roughly balanced Bagging (RBBag) algorithm. 2. C4.5 learner 3. Naive Bayes learner</td>
<td>RBBag combined with C4.5 learner</td>
<td>RBBag combined with C4.5 learner is a better fault predictor than calling structure method.</td>
<td></td>
</tr>
<tr>
<td>ID8</td>
<td>1. Fault detection models used like 1.1 Goel-Okumoto model. 1.2 Yamada delayed S-shaped model. 1.3 Inflected S-shaped model. 2. Fault removal model like 2.1 Non-homogeneous 2.2 Poisson process (NHPP)</td>
<td>Proportion of the leading faults(p), optimal release time.</td>
<td>SRM (software reliability growth models). If leading faults are removed then dependent faults will be removed easily and hence reducing testing cost.</td>
<td></td>
</tr>
<tr>
<td>ID9</td>
<td>The statistical method, machine learning methods, neural network techniques and clustering techniques were applied on below two metrics. 1. Requirement matrix 2. Code matrix</td>
<td>Probability of Detection (PD), Probability of False Alarms (PF) For CMI data set PD = 0.99729, PF = 0.79518 For PC1 Data set PD = 1, PF = 0.99724</td>
<td>The better prediction model is the combination of requirement metric and code metric applied with K means clustering.</td>
<td></td>
</tr>
<tr>
<td>ID12</td>
<td>Below Search Based Techniques were used for developing software fault prediction models. 1. Genetic Programming (GP) 2. Ant Colony Optimization (ACO) 3. Multi-Objective Particle Swarm Optimization (MOPSO) 4. Genetic Algorithms (GA) 5. Artificial Immune Recognition Systems (AIRS) 6. Simulated Annealing - Probabilistic Neural Network (SA-PNN)</td>
<td>Search Based Techniques, Area under the Curve(AUC)</td>
<td>SBT accuracy = 80.75%, AUC = 0.816 For building software fault prediction models SBT and AUC should be used.</td>
<td></td>
</tr>
<tr>
<td>ID13</td>
<td>For fault prediction K-Nearest Neighbor (KNN) and multiple linear regression (MLR) were applied on 1. Six CK metrics 2. Eleven Object oriented metrics.</td>
<td>Mean absolute error and CNeighbours - K (MAE) MAE = 0.675, CNK = 0.273</td>
<td>KNN regression is normally not affected as number of interacting predictors increases and is a better method for defect prediction when applied combined with with (CK+OO metrics).</td>
<td></td>
</tr>
</tbody>
</table>
that some important papers might have been missed in the search process. All the selected papers were extracted and quality assessed by the first author. The quality was reassessed by the second author for maximum number of the selected papers and SR protocol was also reviewed by the second author to avoid any bias in the SR review process.

**DISCUSSIONS**

The result of this SR review suggests that early fault prediction in software is an important aspect of software engineering.

Our SR review was focused to answer the effects of fault prediction on test effort, quality or reliability of software systems. Fault predictions in object oriented systems were discussed. Early detection of faults in software applications helps to save test effort, increase reliability and quality of the software system were discussed. The parameters which are software fault predictors were discussed in the SR. The better methods, models or techniques and their corresponding accuracy measures were studied.

The below data sets were used in different study ids for software fault predictions.

- Design metrics at class level for object oriented (OO) softwares suggested in ID10 were specific to data set publicly available at PROMISE repository [17] and the statistical analysis tools suggested in ID10 were WEKA, IBM-SPSS. While design metrics suggested in ID13 were specific to Bug prediction data set [18].
- Logistic regression suggested in [10, 11] were specific to NASA promise dataset.
- Fault prediction in software systems on the basis of defective or skewed data set suggested in [5, 9] were specific to faulty data set of NASA promise repository.
- Detection and removal of leading faults suggested in [8] were specific to data sets used at [19, 20].
- Ranking of fault prone modules suggested in [7] were based on data set of NASA promise repository.
- Fault prediction depends on the tokens of source codes suggested in [2] were based on data set available on PROMISE repository.
- Product metrics and process metrics suggested in [13] were specific to NASA data set while those metrics suggested in [14] was specific to Metric Repository [21].

**CONCLUSIONS**

The early fault prediction reduces cost of test effort, increases reliability and quality in software systems. Fault prediction is dependent on fault proneness of data set used in software. Identifying and ranking of most fault prone modules helps development managers to take strategic decisions about these modules which saves testing time, cost and delivery time of software. Identifying leading faults and removing them automatically removes dependent faults. This eliminates the need to test the entire software system.

In conclusion our SR review guarantees the below mentioned findings about software fault predictions. Apart from above findings other findings were concluded and addressed in answers to Review Questions. However there is no evidence found regarding the below aspects of fault prediction techniques in our SR review. The data sets used in different study ids were majorly from NASA promise repository. Fault prediction techniques with defective data set, ranking of faulty modules are expected to be tested on other data sets from other domains like biological domain. Although it is evident that faults can be predicted from certain source code tokens but there is no evidence for cross platform fault prediction techniques. In other words how much fault is carried further if same source code tokens being used in different operating systems, different hardware or application interfaces.
It is evident that product metric, process metrics and object oriented metrics are widely used in fault prediction techniques. But fault prediction result is also dependent on human expertise apart from these metrics. So measuring human expertise in software fault prediction techniques is expected for future work. It is evident that fault prediction is dependent on skewed data. But there is no evidence of Fault prediction techniques for big data with real time and interactive data sets in this SR review and is expected for future work.

REFERENCES
