Pertanika J. Sci. & Technol. 30 (3): 2157 - 2171 (2022)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Review Article

Classification of Fault Prediction: A Mapping Study

Sasha Farhana Shamsul Anwar[1], Marshima Mohd Rosli[1,2]* and Nur Atiqah Sia Abdullah[1]

[1]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 UiTM, Shah Alam, Selangor, Malaysia
[2]Institute for Pathology, Laboratory and Forensic Medicine (I-PPerForM), University Teknologi MARA, 47000 UiTM, Sungai Buloh, Selangor, Malaysia

ABSTRACT

Software fault prediction is an important activity in the testing phase of the software development life cycle and involves various statistical and machine learning techniques. These techniques are useful for making accurate predictions to improve software quality. Researchers have used different techniques on different datasets to build fault prediction in software projects, but these techniques vary and are not generalised. As a result, it creates challenges that make it difficult to choose a suitable technique for software fault prediction in a particular context or project. This mapping study focuses on research published from 1997 to 2020 involving fault prediction techniques, intending to determine a classification of fault prediction techniques based on problem types that researchers need to solve. This study conducted a systematic mapping study to structure and categorise the research evidence that has been published in fault prediction. A total of 82 papers are mapped to a classification scheme. This study identified research gaps and specific issues for practitioners, including the need to classify fault prediction techniques according to problem types and to provide a systematic way to identify suitable techniques for fault prediction models.

Keywords: Fault prediction techniques, fault prediction, software metrics, systematic mapping

ARTICLE INFO

Article history:
Received: 20 August 2021
Accepted: 16 December 2021
Published: 25 May 2022

DOI: https://doi.org/10.47836/pjst.30.3.23

E-mail addresses:
sashafarhana95@gmail.com (Sasha Farhana Shamsul Anuar)
marshima@uitm.edu.my (Marshima Mohd Rosli)
atiqah684@uitm.edu.my (Nur Atiqah Sia Abdullah)
* Corresponding author

INTRODUCTION

Software development projects in any organisation worldwide include software quality management strategies and processes to determine the stability of software before

ISSN: 0128-7680
e-ISSN: 2231-8526

the deployment phase (Kassie & Singh, 2020). However, software development projects are mainly a human-driven process, requiring intensive efforts from software engineers to manage requirements, design, source codes, defect data and documentation, and this human-driven process may introduce defects or faults throughout the software development life cycle, which may affect the project management process, particularly concerning project schedules and budget (Illes-Seifert & Paech, 2010; Kim et al., 2008; Rosli et al., 2011).

Software fault prediction is a process of estimating and predicting faults or defects in software modules during a defined period of development and operation. Research in the field of software fault prediction mainly focuses on various methods and tools, such as artificial intelligence and data mining, to detect faults in the software modules (Dejaeger et al., 2013; Geng et al., 2018; Khoshgoftaar et al., 2006; Al Qasem et al., 2020). In general, efforts are concentrated on the number of fault predictions, and system reliability estimation focuses on time failure and understanding the significance of the design and testing process on the defect count (Kastro & Bener, 2008). There are three components commonly involved in software fault prediction: data mining techniques and machine learning algorithms, software characteristics, and fault data.

Software fault prediction reduces cost and improves data quality performance with a characteristics identifier to differentiate and filter the available software modules to continue the next process testing phase (Al Qasem et al., 2020). However, most of the existing prediction models use binary classification that explains whether a source code module or method level is categorised as faulty or non-faulty. Software project managers may, therefore, not differentiate the results of the prediction information regarding the nature and type of fault-prone modules. Thus, it is better to devise a prediction model with actionable results for managers (Caglayan et al., 2015).

Prior review studies reporting on software prediction (Catal, 2011; Catal & Diri, 2009; Murillo-Morera et al., 2015) explored a particular area of fault prediction activities. Recently, Rathore et al. reported a similar review study that examined the main activities in fault prediction, including fault prediction techniques, software metrics, performance measurements and data quality issues (Rathore & Kumar, 2017). They presented a taxonomy of fault prediction techniques based on machine learning and the statistics environment. They found that most studies used object-oriented metrics and process metrics in the fault prediction model.

This research, therefore, presents the outcome of a systematic mapping study of recent research on activities of the software fault prediction process, particularly to obtain information about which techniques are suitable for which form of prediction and what software metrics are used. The mapping study addresses the following research questions: (1) What problem types are contended with in fault prediction models addressed in the literature? (2) What forms of prediction are used in fault prediction models addressed in the literature? (3) What software metrics are used in fault prediction models addressed in the literature?

Researchers propose a variation of robust techniques for different reasons. Some propose techniques concerning how the technique affects the fault prediction results. Others propose a technique to deal with the fault prediction capabilities and a different set of software metrics for improving fault prediction. Both groups are important, but the earlier group that captures the fault prediction model may perform best when the right technique is selected for the appropriate domain of the dataset. This study classifies the papers regarding how researchers use the techniques in software fault prediction.

This study also aims to highlight important research gaps in fault prediction techniques. A total of 82 studies were selected for this mapping study. This study presents the synthesis of evidence for fault prediction techniques based on six classification subcategories: (1) classification, (2) prediction, (3) clustering, (4) optimisation, (5) anomaly detection and (6) rendering. This study also discusses the form of prediction and software metrics used in the studies.

RELATED WORK

Over the past decade, many studies have been published in fault prediction research, but only a few have reported explicitly on the fault prediction techniques. This absence of fault prediction techniques reporting is confirmed by four studies that conducted reviews of fault prediction in software engineering (Catal, 2011; Catal & Diri, 2009; Murillo-Morera et al., 2015; Rathore & Kumar, 2017).

In 2009, Catal and Diri performed a systematic review of fault prediction studies (Catal & Diri, 2009). They identified 74 studies published from 1997 to 2007 that addressed fault prediction activities in the literature. They classified the papers into four categories: (1) types of metrics, (2) types of methods, (3) evaluation criteria and (4) publication types. They mostly assessed the usage of machine learning algorithms, public datasets, and software metrics in fault prediction research. They reported that most of the papers used machine learning algorithms as the preferred method for fault prediction.

In 2011, Catal performed an updated systematic review on fault prediction techniques (Catal, 2011). He found 90 papers published from 1990 to 2009 and classified the papers according to the publication year. He evaluated the performance of fault prediction techniques to predict software fault modules. He reported that most papers applied method-level metrics and machine learning techniques.

In 2015, Murillo-Morera et al. performed a systematic mapping study to categorise fault prediction techniques, metrics and performance metrics to determine patterns and schemes to predict software faults (Murillo-Morera et al., 2015). They found 70 studies published from 2002 to 2014. They classified the papers into three main categories: software metrics, techniques, and models. They proposed six categories of techniques: (1) machine learning, (2) machine learning and classification, (3) machine learning and clustering, (4) machine learning and statistical analysis, (5) clustering and (6) statistical analysis.

In 2017, Rathore et al. conducted a comprehensive review study on software fault prediction activities that included fault prediction techniques, software metrics and performance evaluation (Rathore & Kumar, 2017). They found 78 papers published from 1993 to 2016. They classified the papers into four categories: software metrics, forms of prediction, fault prediction techniques and performance measurements. They proposed four categories of prediction techniques: (1) statistical techniques, (2) supervised learning algorithms, (3) semi-supervised learning algorithms and (4) unsupervised learning algorithms. They reported that several research studies used faulty and non-faulty techniques as the performance measure of fault prediction. However, many studies paid scant attention to performance-based predicting faults and the severity of faults issues.

These review studies focused on a specific dimension or area in fault prediction activities; however, none specifically examined which fault prediction techniques are suitable for a given problem. Such review studies mainly help us recognise the fault prediction techniques discussed in the literature. However, these studies tend to focus on how researchers use fault prediction techniques rather than how they affect their fault prediction model for a given problem. This study conducts a systematic mapping study to determine which fault prediction techniques might affect the prediction model for a specific problem. Although the general goal of the mapping study appears to be similar to past review studies (Catal, 2011; Murillo-Morera et al., 2015; Rathore & Kumar, 2017), this study aims to identify studies that are concerned with how the fault prediction techniques influence the fault prediction model for a particular problem.

METHOD

A systematic mapping study involves a well-defined methodology to identify and review relevant research evidence. Therefore, it provides a clear explanation for specific research questions. Further, these specific research questions must have been used before exploring the topic under investigation. Therefore, the mapping study adopted the guidelines suggested by Peterson et al. (2008), Kitchenham et al. (2011), and Budgen et al. (2008).

This study aimed to categorise and structure empirical evidence in papers that discussed fault prediction techniques. This study will help researchers and practitioners gather, classify, and aggregate results in related studies to identify research gaps and challenges for improvement in the body of knowledge. The systematic mapping process consists of five steps: (1) defining the research questions, (2) searching, (3) selection of papers, (4) abstract keywording, and (5) data extraction and mapping of publications.

Definition of Research Questions

This research formulated the research questions according to the PICOC model by Kitchenham et al. (2011). The research questions are detailed below:

- RQ1: What problem types of fault prediction techniques are dealt with in fault prediction models addressed in the literature?
- RQ2: What forms of prediction are used in fault prediction models addressed in the literature?
- RQ3: What software metrics are used in fault prediction models addressed in the literature?

Conducting Search

A thorough search is required to identify relevant literature. For this reason, a search strategy is necessary, which entails identifying the right search string with the relevant terms and keywords. It will ensure wide coverage and increase the chance of identifying the right publication. Therefore, in this mapping study, the steps used to construct the search string were as follows:

- identify keywords in relevant papers
- search synonyms for identified keywords
- formulate search strings using Boolean OR to include alternative spellings and synonyms and Boolean AND to combine major terms.

The search string was as follows:
("software fault prediction" OR "software defect prediction") AND ("fault prediction techniques" OR "defect prediction techniques" OR "software fault prediction techniques") AND ("software fault datasets" OR "software defect datasets") AND ("systematic mapping") AND ("software metrics") AND ("empirical studies")

Papers from important conferences and journals in the research field, such as *IEEE Transaction of Software Engineering*, *IET Software* and *Journal of Software Engineering,* were included in this process. This study is limited to the fields of computer science and software engineering.

This study examined papers written in the last 23 years, from 1997 to 2020. This decision was based on an earlier study (Rathore & Kumar, 2017) conducted in a systematic study published in 2017. The search method was divided into two parts: primary and secondary searches.

This study used the search string in the primary search on Scopus, ScienceDirect, SpringerLink, ACM Digital and IEEE Xplore online databases. According to the requirements, the above search string was customised for each online database. Table 1 summarises the results of the primary search.

This study used a snowball method in the secondary search to find related papers by reviewing references for the papers chosen in the first part. Then, from the earlier systematic review (Rathore & Kumar, 2017), related papers were selected that were released between 1997 and 2020. With that analysis, a similar search string was used. Table 2 summarises the search results used for a secondary search.

Table 1
Results of primary search

Online database	Search results	Relevant papers
Scopus	191	6
SpringerLink	100	7
ScienceDirect	31	3
ACM Digital Library	23	2
IEEE Xplore	21	4
Total	366	22

Table 2
Results of secondary search

Method	Search results	Relevant papers
Snowball	4	2
Papers from the previous study	78	58
Total	82	60

Selection of Papers

This study defined inclusion and exclusion criteria for this mapping study to avoid a biased selection of papers. The inclusion criteria were (1) peer-review papers that discuss the fault prediction techniques in fault prediction models and (2) peer-review papers that discuss the forms of prediction and software metrics in fault prediction models. The exclusion criterion was peer-review papers that describe fault prediction techniques but not the software metrics.

After performing the automatic search, a set of 366 papers was obtained. The screening process was done by inspecting the title and abstract of each paper to exclude unrelated and identical documents. As a result, it reduced the number of papers in the primary search from 366 to 22 relevant papers. For the secondary search results, basic and detailed inclusion and exclusion criteria were applied for the 82 selected papers, reducing the relevant papers to 60. At this point, the total screening results were reduced to only 82 relevant papers[1].

Developing Classification Schemes

This study developed classification schemes by identifying the keywords and concepts in the abstract of each paper. Next, the keywords and concepts were combined to produce the outline of the terms used in the research. Then, suitable keywords were selected to structure the schemes or categories. Finally, three categories were determined: problem type of prediction (RQ1), a form of prediction (RQ2) and software metrics (RQ3).

Data Extraction And Mapping of Studies

Excel spreadsheet was used for data logging and output production. All the data and

[1] The list of papers included in this study is available from: https://tinyurl.com/SMFaultdata

identified papers were mapped into three categories. Results were tabulated into tables, and publication frequencies were calculated in each category.

RESULTS

Problem Type of Prediction (RQ1)

A classification scheme of fault prediction techniques was constructed into groups of problem types based on the issues that researchers need to solve. The problem types are shown in Table 3.

Table 3
Problem type of fault prediction

Problem type	**Fault prediction techniques**
Classification	Naive Bayes, decision tree, K-nearest neighbour
Prediction	Linear regression, logistic regression
Clustering	Random forest, bagging boosting
Optimisation	Gradient algorithm, batch gradient descent, mini-batch gradient descent, stochastic gradient descent
Anomaly detection	Support vector machine, least squares—support vector machine, kernel
Rendering	Rank boost

The 82 papers were grouped into six categories of problem type for fault prediction techniques: (1) classification, (2) prediction, (3) clustering, (4) optimisation, (5) anomaly detection and (6) rendering. As shown in Table 4, the distribution of papers was not consistent throughout the years 1997–2020. However, there are high numbers of publications in 2012 to solve problems related to classification (seven papers) and clustering (four papers). Classification techniques such as Naive Bayes performed better than more complex predictors and produced good precision among the other statistical techniques. However, Naive Bayes had the lowest recall of the predictor models, and its lack of cross-project prediction is well documented (Hosseini et al., 2016; Peters et al., 2013).

Twelve papers were published in 2016 that solved problems related to clustering and prediction. The clustering techniques included k-means algorithms, X-mean clustering (Seo & Bae, 2013), fuzzy and expectation maximisation (EM) clustering, and density-based spatial clustering (Garcia et al., 2016). In addition, a reasonable number of papers have been published to solve problems with anomaly detection, rendering and prediction. However, the trend of publications decreased for optimisation using gradient algorithms, with five papers throughout the years.

Table 4
Distribution of papers on problem type of fault prediction technique

Year	Classification	Prediction	Clustering	Optimisation	Anomaly detection	Rendering	Total
1997	0	1	0	0	0	0	1
2000	0	1	0	0	0	0	1
2004	4	0	0	0	0	0	4
2005	1	1	0	0	1	0	3
2006	1	0	1	0	0	0	2
2007	5	0	1	0	1	2	9
2008	1	0	0	0	0	1	2
2009	1	0	0	0	0	2	3
2010	0	0	0	1	0	0	1
2011	1	0	1	2	0	1	5
2012	7	1	4	0	5	1	18
2013	2	0	1	1	1	1	6
2014	0	0	2	0	2	1	5
2015	1	0	0	0	1	0	2
2016	1	5	5	0	0	1	12
2020	3	0	2	1	2	0	8
Total	**28**	**9**	**17**	**5**	**13**	**10**	

Form of Prediction (RQ2)

As shown in Figure 1, three types of fault prediction addressed in the literature, and two papers that applied more than one type of fault prediction in their research were found, so the counts sum up to more than 82. In addition, the category ('Did not say') was added

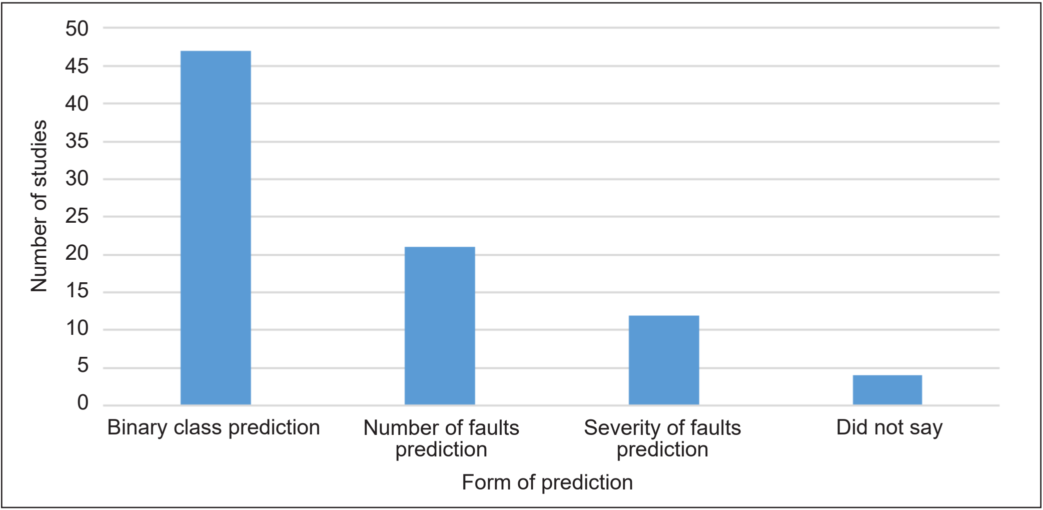

Figure 1. Form of prediction

to group papers that described fault prediction techniques but did not identify the form of prediction.

In this study, most of the published papers (47) used binary class prediction in the fault prediction model, and most of these divided the category into faulty or non-faulty classes. In addition, these studies identified the faulty class as a module that captures more than one fault and the non-faulty class as a module that captures zero faults (Gokhale & Lyu, 1997; Mendes-Moreira et al., 2012; Vandecruys et al., 2008).

As shown in Figure 1, 21 papers used the number of faults prediction, and 12 papers used the severity of faults prediction. Past studies rarely used severity of faults prediction because the available datasets categorised fault severity according to various classifications. However, few studies built fault prediction based on the severity of faults using object-oriented metrics and reported that object-oriented metrics improve fault severity prediction in software projects (Ardil & Sandhu, 2010; Zhou & Leung, 2006).

Software Metrics (RQ3)

This study aimed to identify the software metrics used in fault prediction models. Figure 2 shows the frequency of software metrics used in the selected papers. The software metrics were classified into five categories: traditional metrics, object-oriented metrics, dynamic metrics, process metrics, and did not say.

Traditional metrics (34 papers) were used most frequently in fault prediction models in the literature. In addition, the traditional metrics are used during the beginning of software engineering evolution. This study found four kinds of traditional metrics: size-based (e.g., source lines of code (SLOC), function points (FP), and kilo-SLOC), quality-based (e.g.,

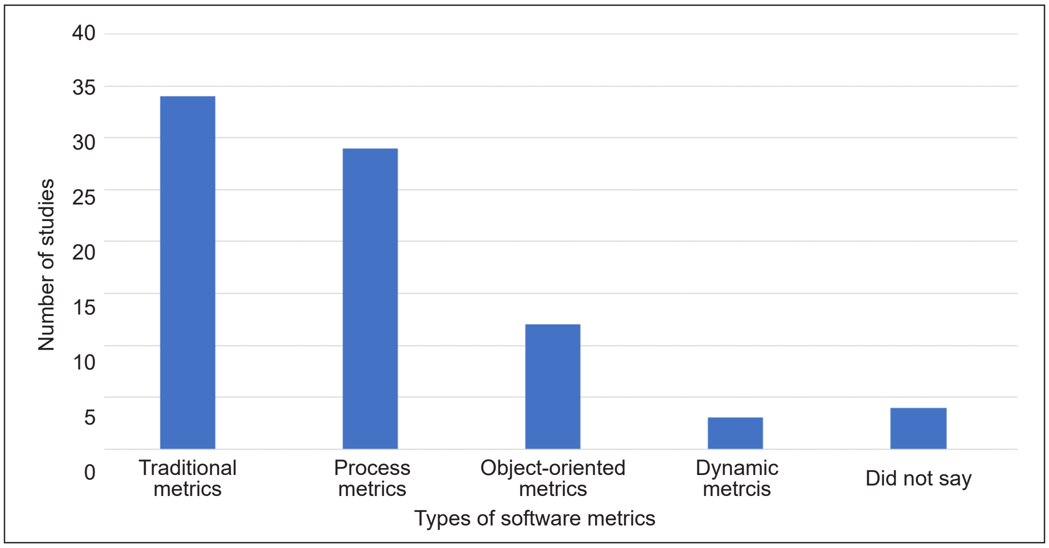

Figure 2. Types of software metrics

defects per function points and defects per source lines of code after delivery), system-complexity (e.g., cyclomatic complexity and McCabe complexity) and Halstead metrics (e.g., a total of distinct operands and a total of distinct operators).

Process metrics are measures of characteristics of the software development process collected across the software development life cycle. These metrics are used to improve software development and maintenance. This study found 29 papers that used process metrics, including code delta, code churn (Shin et al., 2011), change, developer-based and requirement metrics (Yadav & Yadav, 2015). Code delta is the delta of LOC and delta of changes. Code churn mainly involves total LOC, churned LOC, deleted LOC, file count, churn count and files churned. Finally, change metrics are LOC related to revisions, refactoring, bug fixes and authors.

Object-oriented metrics are commonly used in modern software development for fault prediction models. However, this study found only 12 papers that used object-oriented metrics, indicating low usage of object-oriented metrics reported from 1997 to 2020, inconsistent with results from the past study (Murillo-Morera et al., 2015). It may be due to differences in the search string for systematic mapping applied in that study.

Three studies used dynamic metrics in building fault prediction for their software projects. Some of the dynamic metrics are metrics suite (e.g., export object coupling and import object coupling) (Hall et al., 2012), Arisholm metrics suite (e.g. IC_OD and IC_OM) (Shin et al., 2009), and Mitchell metrics suite (e.g. coupling between objects and dynamic coupling between two classes) (Weyuker et al., 2007).

Mapping

Figure 3 presents a distribution map of papers over a defined classification scheme: problem type of prediction, a form of prediction and software metrics. Problem type of prediction is shown on the *y*-axis, a form of prediction - on the left *x*-axis, and software metrics on the right *x*-axis. The size and number of each bubble represent the number of publications in the corresponding category pair. Papers were assembled in more than one classification scheme for each category because they could make several contributions. Therefore, the total paper count does not equal the final total of 82.

The mapping results imply that most of the research effort involved solving problems related to classification using traditional metrics and LOC metrics and applied binary class prediction in the fault prediction model. Furthermore, it indicates that techniques such as naive Bayes, decision tree and K-nearest neighbour might have outperformed other techniques in terms of performance.

Another technique that the researchers widely used is anomaly detection in binary class prediction. This technique was applied mostly with LOC metrics to improve intrusion detection and security attacks (Hosseinzadeh et al., 2021; Khan et al., 2007; Mohammed &

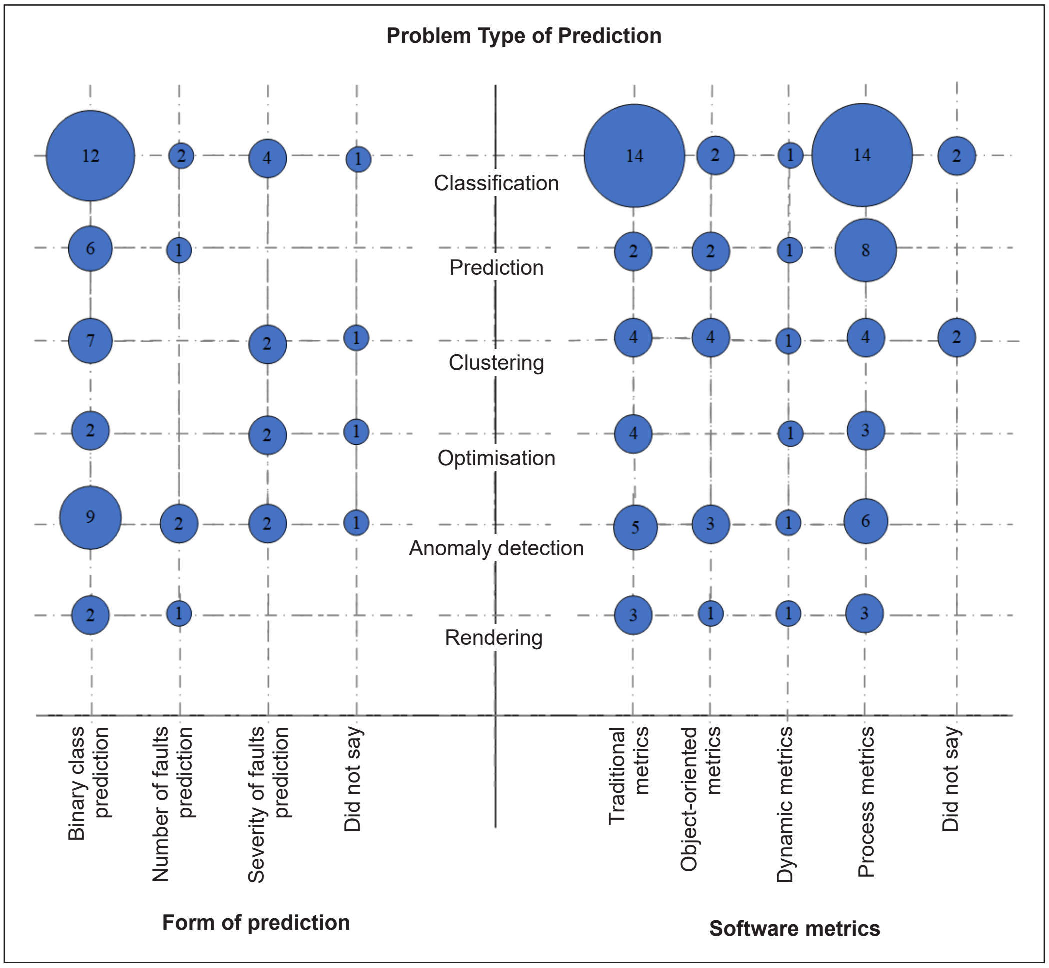

Figure 3. Distribution of research by problem type of prediction, a form of prediction and software metrics

Sulaiman, 2012). The researchers have widely used clustering techniques in binary class prediction with traditional, object-oriented, and LOC metrics. The problem types that received the least attention among researchers were rendering and optimisation.

DISCUSSIONS

The mapping study was designed to investigate current research on the software fault prediction process activities, such as fault types, forms of prediction and software metrics used in fault prediction. This study found 82 papers from a total of 366 potential papers. These papers were classified into three categories: problem type of prediction, a form of prediction and software metrics.

Many papers on fault prediction techniques to solve various problems have been published (Catal & Diri, 2009; Murillo-Morera et al., 2015; Rathore & Kumar, 2017).

However, some fault prediction techniques effectively solve specific problems, and no categories that classify the fault prediction techniques exist. As mentioned earlier, this study constructed a classification scheme for fault prediction techniques based on the issues or problems that needed to be solved. The classification scheme consists of six groups: classification, prediction, clustering, optimisation, anomaly detection and rendering.

These groups are based on the issues or problems that researchers need to solve. Eighty-two relevant papers were classified into these categories and found that these groups are common and useful in choosing a suitable fault prediction technique to solve a particular problem.

The results highlight that binary class prediction is the most popular prediction form used by researchers with various classification prediction techniques. This finding shows that most researchers preferred to perform binary class prediction as it is a less complex and straightforward prediction for a given dataset. These findings are consistent with past studies (Rathore & Kumar, 2017), showing that most publications applied binary class prediction in fault prediction models.

Traditional metrics and LOC have been widely discussed in the literature concerning the many types of software metrics. However, this study found that the number of papers based on object-oriented and dynamic metrics is relatively low. This finding shows that various factors may influence researchers to select suitable software metrics for fault prediction techniques.

The validity threat in this mapping study is the inappropriate selection of publications due to inadequate search terms in the search string. This study has refined the search string and performed search testing before executing the search strategy process to mitigate the threat. In addition, the search string was verified by another researcher to locate any missing terms in the search string.

CONCLUSION

This mapping study aimed to determine a classification scheme for fault types, forms of prediction and software metrics applied in fault prediction models. The results of this mapping study have revealed that some fault prediction techniques are effective in solving specific kinds of problems. Hence, there is a need to construct categories of fault prediction techniques and have taken the first step by proposing a classification scheme for fault prediction techniques based on the issues or problems needed to be solved.

The authors are currently constructing a model for building software fault prediction by applying the classification scheme of fault prediction techniques. In future, the authors plan to develop a recommender system to recommend the most suitable fault prediction technique to solve a fault prediction problem.

ACKNOWLEDGEMENT

The authors want to thank the Universiti Teknologi MARA and the Ministry of Education Malaysia for their financial support of this project under FRGS Grant No. 600-IRMI/FRGS 5/3 (212/2019).

REFERENCES

Al Qasem, O., Akour, M., & Alenezi, M. (2020). The influence of deep learning algorithms factors in software fault prediction. *IEEE Access*, *8*, 63945-63960. https://doi.org/10.1109/ACCESS.2020.2985290

Ardil, E., & Sandhu, P. S. (2010). A soft computing approach for modeling of severity of faults in software systems. *International Journal of Physical Sciences*, *5*(2), 74-85. https://doi.org/10.5897/IJPS.9000037

Budgen, D., Turner, M., Brereton, P., & Kitchenham, B. (2008, September 10-12). Using mapping studies in software engineering. In *Proceedings of Psychology of Programming Interest Group Workshop* (Vol. 8, pp. 195-204). Lancaster, UK.

Caglayan, B., Misirli, A. T., Bener, A. B., & Miranskyy, A. (2015). Predicting defective modules in different test phases. *Software Quality Journal*, *23*(2), 205-227. https://doi.org/10.1007/s11219-014-9230-x

Catal, C. (2011). Software fault prediction: A literature review and current trends. *Expert Systems with Applications*, *38*(4), 4626-4636. https://doi.org/10.1016/j.eswa.2010.10.024

Catal, C., & Diri, B. (2009). A systematic review of software fault prediction studies. *Expert Systems with Applications*, *36*(4), 7346-7354. https://doi.org/10.1016/j.eswa.2008.10.027

Dejaeger, K., Verbraken, T., & Baesens, B. (2013). Toward comprehensible software fault prediction models using bayesian network classifiers. *IEEE Transactions on Software Engineering*, *39*(2), 237-257. https://doi.org/10.1109/TSE.2012.20

Garcia, L. P. F., de Carvalho, A. C. P. L. F., & Lorena, A. C. (2016). Noise detection in the meta-learning level. *Neurocomputing*, *176*, 14-25. https://doi.org/10.1016/j.neucom.2014.12.100

Geng, R., Wang, X., Ye, N., & Liu, J. (2018). A fault prediction algorithm based on rough sets and back propagation neural network for vehicular networks. *IEEE Access*, *6*, 74984-74992. https://doi.org/10.1109/ACCESS.2018.2881890

Gokhale, S. S., & Lyu, M. R. (1997). Regression tree modeling for the prediction of software quality. In *Proceedings of the Third ISSAT International Conference on Reliability and Quality in Design* (pp. 31-36). International Society of Science and Applied Technologies.

Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A systematic literature review on fault prediction performance in software engineering. *IEEE Transactions on Software Engineering*, *38*(6), 1276-1304. https://doi.org/10.1109/TSE.2011.103

Hosseini, S., Turhan, B., & Mäntylä, M. (2016). Search based training data selection for cross project defect prediction. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering* (pp. 1-10). ACM Publishing. https://doi.org/10.1145/2972958.2972964

Hosseinzadeh, M., Rahmani, A. M., Vo, B., Bidaki, M., Masdari, M., & Zangakani, M. (2021). Improving security using SVM-based anomaly detection: Issues and challenges. *Soft Computing*, *25*(4), 3195-3223. https://doi.org/10.1007/s00500-020-05373-x

Illes-Seifert, T., & Paech, B. (2010). Exploring the relationship of a file's history and its fault-proneness: An empirical method and its application to open source programs. *Information and Software Technology*, *52*(5), 539-558. https://doi.org/10.1016/j.infsof.2009.11.010

Kassie, N. B., & Singh, J. (2020). A study on software quality factors and metrics to enhance software quality assurance. *International Journal of Productivity and Quality Management*, *29*(1), 24-44. https://doi.org/10.1504/IJPQM.2020.104547

Kastro, Y., & Bener, A. B. (2008). A defect prediction method for software versioning. *Software Quality Journal*, *16*(4), 543-562. https://doi.org/10.1007/s11219-008-9053-8

Khan, L., Awad, M., & Thuraisingham, B. (2007). A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB Journal*, *16*(4), 507-521. https://doi.org/10.1007/s00778-006-0002-5

Khoshgoftaar, T. M., Seliya, N., & Sundaresh, N. (2006). An empirical study of predicting software faults with case-based reasoning. *Software Quality Journal*, *14*(2), 85-111. https://doi.org/10.1007/s11219-006-7597-z

Kim, S., Whitehead, E., & Zhang, Y. (2008). Classifying software changes: Clean or buggy? *IEEE Transactions on Software Engineering*, *34*(2), 181-196. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4408585

Kitchenham, B. A., Budgen, D., & Brereton, O. P. (2011). Using mapping studies as the basis for further research - A participant-observer case study. *Information and Software Technology*, *53*(6), 638-651. https://doi.org/10.1016/j.infsof.2010.12.011

Mendes-Moreira, J., Soares, C., Jorge, A. M., & de Sousa, J. F. (2012). Ensemble approaches for regression. *ACM Computing Surveys*, *45*(1), 1-40. https://doi.org/10.1145/2379776.2379786

Mohammed, M. N., & Sulaiman, N. (2012). Intrusion detection system based on SVM for WLAN. *Procedia Technology*, *1*, 313-317. https://doi.org/10.1016/j.protcy.2012.02.066

Murillo-Morera, J., Quesada-López, C., & Jenkins, M. (2015, April 22-24). Software fault prediction: A systematic mapping study. In *CIBSE 2015 - XVIII Ibero-American Conference on Software Engineering* (pp. 446-459). Lima, Peru.

Peters, F., Menzies, T., & Marcus, A. (2013). Better cross company defect prediction. In *2013 10th Working Conference on Mining Software Repositories (MSR)* (pp. 409-418). IEEE Publishing. https://doi.org/10.1109/MSR.2013.6624057

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008, June 26-27). Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering, EASE 2008* (pp. 1-10). University of Bari, Italy. https://doi.org/10.14236/ewic/EASE2008.8

Rathore, S. S., & Kumar, S. (2017). A study on software fault prediction techniques. *Artificial Intelligence Review*, *51*(2), 255-327. https://doi.org/10.1007/s10462-017-9563-5

Rosli, M. M., Teo, N. H. I., Yusop, N. S. M., & Mohammad, N. S. (2011). The design of a software fault prone application using evolutionary algorithm. In *2011 IEEE Conference on Open Systems* (pp. 338-343). IEEE Publishing. https://doi.org/10.1109/ICOS.2011.6079246

Seo, Y. S., & Bae, D. H. (2013). On the value of outlier elimination on software effort estimation research. *Empirical Software Engineering*, *18*(4), 659-698. https://doi.org/10.1007/s10664-012-9207-y

Shin, Y., Bell, R., Ostrand, T., & Weyuker, E. (2009). Does calling structure information improve the accuracy of fault prediction? In *2009 6th IEEE International Working Conference on Mining Software Repositories* (pp. 61-70). IEEE Publishing. https://doi.org/10.1109/MSR.2009.5069481

Shin, Y., Meneely, A., Williams, L., & Osborne, J. A. (2011). Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. *IEEE Transactions on Software Engineering*, *37*(6), 772-787. https://doi.org/10.1109/TSE.2010.81

Vandecruys, O., Martens, D., Baesens, B., Mues, C., De Backer, M., & Haesen, R. (2008). Mining software repositories for comprehensible software fault prediction models. *Journal of Systems and Software*, *81*(5), 823-839. https://doi.org/10.1016/j.jss.2007.07.034

Weyuker, E. J., Ostrand, T. J., & Bell, R. M. (2007). Using developer information as a factor for fault prediction. In *Third International Workshop on Predictor Models in Software Engineering (PROMISE'07: ICSE Workshops 2007)* (pp. 8-8). IEEE Publishing. https://doi.org/10.1109/PROMISE.2007.14

Yadav, H. B., & Yadav, D. K. (2015). A fuzzy logic based approach for phase-wise software defects prediction using software metrics. *Information and Software Technology*, *63*, 44-57. https://doi.org/10.1016/j.infsof.2015.03.001

Zhou, Y., & Leung, H. (2006). Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Transactions on Software Engineering*, *32*(10), 771-789. https://doi.org/10.1109/TSE.2006.102