Software Risk Analysis with the use of Classification Techniques: A Review

Muhammad Naeem Ahmed Khan
Independent Researcher
Islamabad, Pakistan
mnak2010@gmail.com

Aamir Mehmood Mirza
Department of Computer Science
Baluchistan University of Information Technology, Engineering and Management Sciences, Quetta, Pakistan
mirza.aamir@buitms.edu.pk

Imran Saleem
School of Professional Advancement
University of Management and Technology
Lahore, Pakistan
imran.saleem@umt.edu.pk

***Abstract*—Risk analysis and management is a critical aspect of the software development process. Various risks are associated with every phase of the software development lifecycle. The early identification of risks in each phase of software development coupled with mitigating plans can help to reduce the cost of the product and increase software quality. This study aims to explore various tools and techniques used in the literature of analyzing and managing risks. Most risk analysis techniques have been applied in the requirement analysis phase, so there is a scarcity of tools supporting automated risk analysis. Accommodating various types of risk factors to predict the software risks reduces the accuracy of the classifier.**

Keywords-risk analysis; software risk; classification techniques

I. INTRODUCTION

A software risk is a threat that could have a negative effect on software quality, delay the project or exceed the budget. It is necessary to identify various types of risk to a software project during the early phases of Software Development Life Cycle (SDLC). Software risk poses serious threats to the software quality as the desired functionality may be compromised. The specific areas of SDLC which are affected by risks are software development, testing, and maintenance phases. Software quality can be ensured by forecasting the risks that an existing software project can face. For this purpose, we use defect prevention, defect reduction, and defect containment. Defect prediction is necessary to forecast the presence of defects in software modules. For defect prediction, we use historical data of the past projects and assess how many software defects are likely to present in the current projects. The purpose of this study is to find the merits and the demerits of the different classification techniques that have been applied for defect prediction. A critical review showing the usefulness of the various classification techniques on different types of software project datasets has been conducted in this report.

Developing an efficient software fault/risk prediction method is a highly demanding challenge. Software fault/risk prediction plays an important role in analyzing software quality and balancing software development cost. Early and accurate prediction of software risk has become a critical issue to project success. Many researchers have proposed different prediction approaches such as an Artificial Neural Networks, Fuzzy Logic, Decision Tree, Bayesian Network (BN), etc. to overcome the problem of risk in software development. As the demand for good software quality is increasing, increased software size and complexity may lead to increasing software faults which require identification at the early stages of SDLC.

II. LITERATURE REVIEW

Authors in [1] address the way risks affect different software development efforts. The study explores the causes of different risks with the use of a BN. The study combines expert knowledge and V-structure discovery algorithm to generate a BN that performs causality analysis to manage software project-related risks and to improve prediction accuracy. Causality analysis is more useful than correlated analysis as the later approach does not help in effective risk planning. The objective of the study is to combine risk analysis and risk control for effective software risk management. Authors employ expert knowledge to build a Bayesian network in order to identify risks. The study is focused on three critical risk factors, namely inadequate requirements, lack of user cooperation, and poor project planning. However, 27 equally important risk factors were identified and Bayesian Network is efficient only for small datasets. Involving many variables makes it more complex and different to understand.

The way risks affect the analysis and design stages of SDLC is addressed in [2], in which a methodology called State-Based Risk Assessment approach is explored, which is used to estimate risks for different states of a component and estimates the risk for the overall scenario on the basis of the entire component. The study introduced an inter-component State Dependence Graph that is used to estimate the complexity and severity of risk assessment. At first, complexity estimates, for the state of the component within the system, are made. Then, the severity for the component within the scenario is decided using the three hazard technique: functional failure analysis, software failure mode and effect analysis, and software fault tree analysis. The objective of the study is to estimate the overall system risks that are based on the scenario risk and the State Collaboration Test Model of the System (SCOTEM).

Corresponding author: Aamir Mehmood Mirza

Authors in [3] address how risk analysis and planning are difficult to manage effectively. The authors explore an empirical integration of intelligent decision support model for software project risk analysis and planning. The study has two main contributions: An Integration Framework for Intelligent Software Risk Planning (IF-ISPRP) and for minimizing the impact of project risks and a method called MMAKD (Many to Many Actionable Knowledge Discovery) for complex risk planning. Both the framework and the method are integrated to support effectively control software project-related risks. The objective of the study is to combine risk analysis and risk planning modules to control risks. The study assumes that different risk actions can be executed simultaneously where practically this is not possible as risks are managed in some order.

Authors in [4] explore different types and categories of risk and their effects on the software project. The authors elaborate on different aspects and characteristics of risk systematically. They also suggest a software tool for risk assessment for supporting better and quicker decisions. The study combines different types of risk and categories of risk with their characteristics. A tool called RISK is used for determining how an event occurs and to evaluate what consequences it can have on project performance. The study explores risk analysis, focusing on quantitative risk analysis, for making a better decision of what kind of risk has occurred. The objective of the study is to identify different types of risks and categorize them. The way project cost and duration are making the budget estimation difficult for software risk management when trying to develop a high-quality software system is addressed in [5]. The study improves software process risk management by using the software process model with risk management and cost. The study proposes a measurement model that combines process risk management and trustworthiness metrics. The study suggests two metrics, one for software risk management performance and another for trustworthiness that help the project manager to control risks for every software development process. The objective of the study is to improve software risk management, enhancing trustworthiness by avoiding cost overrun and delays in software development and also, to use cost control modules that help improving the software risk management process.

Authors in [6] explored the field of risk management using a knowledge management conceptual framework, called Knowledge-Based Risk Management for identifying how knowledge management helps improving effectiveness. Timely identification of risk helps improving the overall implementation and the quality of the product. Handling risks early in software development helps to reduce the cost of the product while minimizing delays. The study suggests integrating knowledge management activities for risk management and for this purpose, a framework is proposed. Risk occurrence at the early stages of software development and its effect on the success of the final product is explored in [7]. To achieve quality software products in the shortest time, the Goal-Driven Software Development Risk Management Model (GSRM) model is presented that is used to identify and analyze goals, risk, and treatment actions at the early stages of the SDLC. The study presents a layer-based modeling framework that supports software development risk management. It performs activities by using suitable tasks, methods and techniques. The framework is categorized into five dimensions. The objective of the study is to explore risk at early SDLC by using a method called Goal-Driven Software Development Risk Management (GSRM), that performs a treatment action at the early stages of the process, particularly in the requirement phase.

The way faults affect software quality and cost is addressed in [8]. The study combines fuzzy measures to generate concepts connected to the fuzzy integral to achieve better prediction performance of the software module. The objective of the study is to minimize cost and improve the software development process by early prediction of fault-prone software. The authors employ object-oriented metrics and method level metrics for constructing a model that establishes the relationship between software metrics and fault-prones and does not require expert knowledge. Using fuzzy integral for software fault prediction provides significant advantages due to its ability. The major limitation of the proposed approach is that it is computationally extensive and building a model using this technique requires a lot of effort.

Author in [9] addresses how risky software projects affect the overall project success and software risk results, schedule and cost overload. The study explores accuracy evaluation criteria and performance charts to predict whether the project is risky by using a three-layered neural network with back propagation. The study proposes an algorithm which can understand complex patterns with a three-layered neural network architecture. The purpose of using complex datasets is to build a predictive model for the risky nature of projects. The objective of the study is to control risk at an early stage of the software development and minimize cost and time of software construction, while improving the performance of the project. Authors in [10] explore different classification approaches for the purpose of software defect prediction. The objective of the study is to create a cost-sensitive Artificial Neural Network (ANN) by using the ABC algorithm for efficient software defect prediction. The study suggests the Expected Cost of Misclassification (ECM) technique for converting an ANN into a cost-sensitive learner. The study utilizes N-fold cross-validation technique to determine the performance of the proposed classifier. Authors divide the performance of classifiers into two subsections, regardless and regarding cost sensitivity. Both metrics are used for comparison with other studies. The study proposes a classifier for the software defect prediction problem.

Authors in [11] explore how the estimation of risk at the early stages of software development affects reliability and software cost. The study suggests a new fuzzy rule base algorithm for the validation of the proposed fuzzy rule base model while data of 20 actual software projects have been used. The objective of the study is to predict faults at the early stages of SDLC, i.e. requirement analysis phase or design phase for developing highly desirable software with optimal cost. Authors in [12] address how risk and potential failure affect systems, products, and processes. The study suggests that Failure Mode and Effect Analysis (FMEA) can identify failures

in a system by a Risk Priority Number (RPN) which lacks some deficiency and affects effectiveness. The objective of the study is to identify potential failure modes, evaluate their effects and causes, and find the solution to eliminate or reduce the occurrence of failures by using the proposed TOPSIS approach. FMEA techniques lack effectiveness because two different failure modes may have the same RPN but could have different risk implications, and FMEA doesn't differentiate between risk implications. To overcome this issue the TOSIS method has been applied to properly manage the risk. TOPSIS is based on linguistic terms with their rating of failure modes. The limitation of the proposed approach is that it involves both subjective and objective analysis.

Author in [13] explores how the fault proneness affects software quality. A logistic regression method and six machine learning methods (Decision Tree, Support Vector Machine, Group Method of Data Handling Method, Artificial Neural Network, Cascade Correlation Network, and Gene Expression Programming) are analyzed and compared in order to find the relationship between static code metrics and fault proneness. The objective of the study is to develop good quality software by using machine learning methods for analyzing and predicting faults in the early phases of SDLC. The author compared the predictive facility of the models by using Area Under Curve (AUC) that measures Receiver Operating Characteristic (ROC). The results of LR and Machine Learning (ML) methods were compared in the use of two datasets. As a result, the ML method was found to perform better than LR.

Authors in [14] address the way defects affect software maintenance, accuracy, trustworthiness, and evolution. The study explores a defect prediction classification model to reduce the defects that affect the performance of a project or product. This study finds an extension of the ordinal association to relational association rules that predict whether the software module is or not defective and performs an experiment to evaluate the proposed classification model, which is based on relational association rule mining and compares it with other software detection approaches. The objective is to build defect-free high-quality software by identifying and predicting defects.

Authors in [15] address how a two-way decision method causes high misclassification error rates and costs. The study suggests three-way decisions to overcome the problem of two-way decision that includes two kinds of actions based on software defect prediction. In each experiment, 10-fold cross-validation has been performed to show that the proposed method reduces errors and cost. Authors in [16] proposed the TGR model to define risk features in requirement engineering, based on the original Tropos and risk analysis methodologies. The approach encourages the development of numerous cost-based, risk-based and goal-based solutions. This analysis could find the best solution that meets the demands of lower costs and risks. The study also implemented a prototype application to prove the concept. The application takes as input various goals, events and responses and produces various combinations of solutions to enable end-users to select the best one.

Risk management in Open Source Software (OSS) is addressed in [17]. One of the key technologies for achieving shorter time-to-market and better quality of the software system is the reuse of software components from third-party vendors. Such modules, also known as Off-the-Shelf components, come in two types: Commercial-Off-The-Shelf and OSS components. To make an effective use of OSS components, it is important to find how to change the development processes and methods. The study suggests many measures that may be implemented for process improvement and risk reduction depending on the Open-Source Software components, though not all are necessary for a project. Machine learning techniques are employed to build a model predicting potentially faulty modules with respect to their metric data within a given set of software modules [18]. To that end, the Naive Bayesian Classification is used as authors have argued that in many complex real-world situations, Naive Bayesian performs better than techniques such as decision trees. The data sets used in the experiments are organized into two categories, i.e. learning datasets and prediction datasets. Risk factors are analyzed using the proposed model to enhance the risk assessment process.

Risk management has also been used in project management. The main aim of the authors in [19] was to understand the principles of risk assessment in order to establish a model for risk management of IT projects. The proposed conceptual model is based on a synthesis of the PMBOK guide and the risk management framework of PMI by incorporating different areas of expertise extracted from these manuals. Integrating these principles allows the early start of the risk management phase in the project by including key stakeholders in the lifecycle, regularly assessing project threats during the project lifecycle, and creating risk reduction strategies to better match projects with the company's strategy. The effects of uncertainty on the project and the risk event as a consequence of uncertainty are analyzed in [20]. The uncertainty index is proposed as a quantitative measure for assessing project uncertainty. This is done by using entropy as an indicator of system disorder and lack of information. Using this index, the uncertainty of each activity and its increase due to risk effects and changes in project uncertainty as a function of time can be assessed. The proposed solution is implemented and analyzed as a case study. The results can be useful for project managers and other stakeholders in selecting the most effective methods for controlling uncertainty and risk management.

III. CRITICAL EVALUATION

A critical analysis of selected studies is provided in Table I.

IV. CONCLUSION

In this study, we reviewed the latest research pertaining to software risk assessment, software fault prediction, and risk management. The aim of the study was to discover various classifiers that have been tried for defect prediction. Commonly used classification techniques are ANN, ABC algorithm, fuzzy integral, fuzzy hybrid TOPSIS approach, BN, fuzzy rule base model, etc. For classification, various types of software project datasets have been used. It is worth mentioning that some researchers combined risk analysis and risk control mechanisms for effective software risk management

TABLE I. CRITICAL EVALUATION OF SELECTED STUDIES

Ref	Area of focus	Model and tools used	Algorithms and datasets used	SDLC phases covered	Risk dimensions	Risk categories and factors identified	Validation parameters
[6]	Software project risk analysis and management	Probabilistic model	BN with following causality constraints: 1. Expert knowledge 2.V-Structure discovery	Requirement	- Organization environment - User risks - Requirement risks - Project complexity - Planning and control - Team test	Categories: 6 Factors: 27	Accuracy (10-fold cross-validation)
[10]	RM process and integrating knowledge management activities for RM	A conceptual framework called KBRM. KM processes and risk response planning to mitigate risks.	Knowledge-Based Risk Management Model (KBRM)	All	Risk response planning	Categories: 1	Improves project effectiveness
[16]	Risk on analysis of SDLC	A state-based risk assessment approach. SCOTEM	Functional failure analysis, software failure mode and effect analysis, software fault tree analysis	Design	Scenario risks, risks at analysis and design stages	Categories: 2	Experiments to prove efficiency and comparison with existing methods.
[11]	Manageability of risk analysis and planning.	An integrative framework (IF-ISPRP). MMAKD	-	Analysis and planning phase	- Organization environment. - User risks. - Requirement risks. - Project complexity. - Planning and control. - Team test.	Categories: 6 Factors: 27	Includes a number of cost minimal risk cost actions.
[14]	Risks at the early stages of the software development process.	- GSRM. - Layer based modeling framework	-	Early stages	Risk treatment actions at early stages.	Categories: 1 Factors: 17	Explore risks to build a high-quality software product.
[9]	Different risk types and categories and their effects on a project.	Software tool RISK for supporting better and quick decisions.	-	All	- Qualitative risk analysis - Quantitative risk analysis The study focuses on quantitative risk analysis.	Categories: 2	Developes and highlights the importance of automated risk analysis software.
[3]	Software risk management to produce high-quality software	- Software process model with risk management and cost. - Measurement model.	- Trustworthy software process - Software development process - Product trustworthiness	All	Calculations of software risk management performance and trustworthiness values	Categories: 2	Control risk for every subprocess of the software development process.
[1]	Fault-prone effect software quality and software cost.	- Fuzzy integral - Object-oriented metrics - Method-level metrics	- Fuzzy integral classifier. - NASA provided PROMISE and data program repository	All	Predicting fault-prone software	Categories: 1	Minimizes cost and improves software development process by early prediction of fault-prone software.
[7]	Software risk result schedule and cost overload	ANN	OMRON dataset	Early stages	Requirement, estimation, planning, team organization, and project management	Categories: 5 Factors: 22	100% accuracy with the NN model and 87.5% when using logistic regression.
[8]	Software defect prediction issues affecting SDLC	ANN, ECM	ABC algorithm 5 datasets	All	Prediction software defect/risk	Categories: 1	N-fold cross-validation technique
[4]	Risk estimation at the early stages of development affecting reliability and cost	Fuzzy rule based model	Fuzzy rule based algorithm	Requirement phases. Analysis phase and design phase	Requirement fault density, requirement stability, review, inspection, walkthrough, experience of requirement team	Categories: 4	Performance of fault prediction
[5]	Risk and potential failure affect systems, products, processes	TOPSIS	FMEA	All	Potential failure modes.	Categories: 1	Eliminate or reduce failure occurrence.
[15]	Fault proneness affects software quality	- Logistic regression - 6 ML methods - AUC	D T, SVM, G M DHM, ANN, CCN, and GEP. AR1 and AR6 datasets	All	Static code metric and fault proneness	Categories: 2	10-fold cross-validation

[17]	Defects affect software maintenance, accuracy, trustworthiness, and evolution.	-Ordinal association. -Relational association rules.	-DPRAR -NASA datasets	All	Predicting software defects/faults	Categories: 1	Accuracy (cross validation)
[12]	Three-way decision	- Two-way decision method. - Three-way decision method	- 11 NASA datasets. - Six algorithms for comparison.	All	- Defect prone modules. - Non defect prone modules. - Deferment modules.	Categories: 3	10-fold cross-validation
[13]	Risk assessment in requirement engineering	TGR	-	Requirement	Cost, risk prioritization, cost risk analysis	Categories: several	Observations made in terms of cost and cost and risk. Risk-based priority of candidate solutions.
[2]	Risk assessment for OSS	Risk breakdown structure	-	All	Quantitative risk analysis matrix	Categories: 5 Factors: 12	Risk response actions for OSS.
[18]	Analyze risk factors and enhance the risk assessment process	Naïve Bayes classification	Naïve Bayes algorithm	All	Potentially defected modules within a given set of software modules	Categories: 1	85.77 % accuracy when using Naïve Bayes and 82.81% when using NNs.

REFERENCES

[1] Y. Hu, X. Zhang, E. W. T. Ngai, R. Cai, M. Liu, "Software project risk analysis using Bayesian Networks with causality constraints", Decision Support Systems, Vol. 56, pp. 439–449, 2013

[2] M. Ray, D. P. Mohapatra, "Risk analysis: a guiding force in the improvement of testing", IET Software, Vol. 7, No. 1, pp. 29–46, 2013

[3] Y. Hu, J. Du, X. Zhang, X. Hao, E. W. T. Ngai, M. Fan, M. Liu, "An integrative framework for intelligent software project risk planning", Decision Support Systems, Vol. 55, No. 4, pp. 927–937, 2013

[4] Z. Kremljak, C. Kafol, "Types of risk in a system engineering environment and software tools for risk analysis", Procedia Engineering, Vol. 69, pp. 177–183, 2014

[5] J. Li, M. Li, D. Wu, H. Song, "An integrated risk measurement and optimization model for trustworthy software process management", Information Sciences, Vol. 191, pp. 47–60, 2012

[6] S. Alhawari, L. Karadsheh, A. N. Talet, E. Mansour, "Knowledge-based risk management framework for information technology project", International Journal of Information Management, Vol. 32, No. 1, pp. 50–65, 2012

[7] S. Islam, H. Mouratidis, E. R. Weippl, "An empirical study on the implementation and evaluation of a goal-driven software development risk management model", Information and Software Technology, Vol. 56, No. 2, pp. 117–133, 2014

[8] C. Jin, S. W. Jin, "Applications of fuzzy integrals for predicting software fault-prone", Journal of Intelligent Fuzzy Systems, Vol. 26, No. 2, pp. 721–729, 2014

[9] W. M. Han, "Discriminating risky software project using neural networks", Computer Standard & Interfaces, Vol. 40, pp. 15–22, 2015

[10] O. F. Arar, K. Ayan, "Software defect prediction using cost-sensitive neural network", Applied Soft Computing, Vol. 33, pp. 263–277, 2015

[11] S. Chatterjee, B. Maji, "A new fuzzy rule based algorithm for estimating software faults in early phase of development", Soft Computing, Vol. 20, No. 10, pp. 4023–4035, 2016

[12] H. C. Liu, J. X. You, M. M. Shan, L. N. Shao, "Failure mode and effects analysis using intuitionistic fuzzy hybrid TOPSIS approach", Soft Computing, Vol. 19, No. 4, pp. 1085–1098, 2015

[13] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules", Applied Soft Computing, Vol. 21, pp. 286–297, 2014

[14] G. Czibula, Z. Marian, I. G. Czibula, "Software defect prediction using relational association rule mining", Information Sciences, Vol. 264, pp. 260–278, 2014

[15] W. Li, Z. Huang, Q. Li, "Three-way decisions based software defect prediction", Knowlegde-Based Systems, Vol. 91, pp. 263–274, 2016

[16] S. N. Bhukya, S. Pabboju, "Software engineering: Risk features in requirement engineering", Cluster Computing, Vol. 22, No. S6, pp. 14789–14801, 2019

[17] N. D. Linh, P. D. Hung, V. T. Diep, T. D. Tung, "Risk management in projects based on Open-Source Software", 8th International Conference on Software and Computer Applications, Penang, Malaysia, February, 2019

[18] K. Suresh, R. Dillibabu, "Designing a machine learning based software risk assessment model using Naïve Bayes algorithm", TAGA Journal, Vol. 14, pp. 3141--3147, 2018.

[19] A. E. Yamami, S. Ahriz, K. Mansouri, M. Qbadou, E. Illoussamen, "Representing IT projects risk management best practices as a metamodel", Engineering, Technology & Applied Science Research, Vol. 7, No. 5, pp. 2062-2067, 2017

[20] A. Chenarani, E. A. Druzhinin, "A quantitative measure for evaluating project uncertainty under variation and risk effects", Engineering, Technology & Applied Science Research, Vol. 7, No. 5, pp. 2083-2088, 2017