2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)

Software Risk Estimation Through Bug Reports Analysis and Bug-fix Time Predictions

Hussain Mahfoodh
Department of Computer Science
University of Bahrain
Sakheer, Bahrain
20073370@stu.uob.edu.bh

Qasem Obediat
Department of Computer Science
University of Bahrain
Sakheer, Bahrain
qobeidat@uob.edu.bh

***Abstract*—Categorizing the level of software risk components is very important for software developers. This categorization allows the developers to increase software availability, security, and provide better project management process. This research proposes a novel approach risk estimation system that aims to help software internal stakeholders to evaluate the currently existing software risk by predicting a quantitative software risk value. This risk value is estimated using the earlier software bugs reports based on a comparison between current and upcoming bug-fix time, duplicated bugs records, and the software component priority level. The risk value is retrieved by using a machine learning on a Mozilla Core dataset (Networking: HTTP software component) using Tensorflow tool to predict a risk level value for specific software bugs. The total risk results ranged from 27.4% to 84% with maximum bug-fix time prediction accuracy of 35%. Also, the result showed a strong relationship for the risk values obtained from the bug-fix time prediction and showed a low relationship with the risk values from the duplicated bug records.**

Keywords— risk estimation, bug reports, bug-fix time, prediction, risk analysis, risk management

I. INTRODUCTION

Risk management in software development projects is a very important strategy to produce high-quality software [1]. Risk management must be considered in the earliest phases of the software development process. Thus, this risk management allows developers to prevent failures in the software project by detecting bugs [2] that could lead to vulnerabilities threatening the software technical assets. Applying early risk management approaches can also help in a better change and patch management processes [3]. Moreover, software quality [4] and software assurance [5] objectives can be reached for project managers and internal stakeholders to earn the trust of the software users, which guarantees software trustworthiness and its efficiency in the long term. Several studies show the various attributes affecting the process of Software bug prediction methodologies [6, 7]. These methods are needed to assist in making quantitative risk management decisions. The bug-fix time also plays an important role in making risk decisions. A project manager wants to know the amount of time needed for developers to fix a certain bug [8], especially if the bug or the error has a high negative impact on the system. Therefore, it is useful for the project manager to focus on what matters first and to have a prior idea on which resources and tasks should be handled and delegated for.

The appearances of duplicated bugs that have not been fixed within the software component are also a problem to be considered. The appearance of the same threatening bug across multiple software components can have a severe impact on system confidentiality, availability, and integrity. System owners could use the findings of a numeric risk value for the duplicated bugs category to tackle the urgency of the mitigation plans needed to be implemented early.

A machine learning threat-based approach is needed to estimate software risk. Multiple software risk estimation studies used historical reports to estimate software deliverable timelines [9] or prioritize requirements [10]. Different studies analyzed external factors affecting software such as technical configurations, user activity, manual bug reports tracking [11], and change history [12]. Another study focused on risk estimation from a project-level perspective with considering team knowledge level, requirement clarity, and financial feasibility through a probabilistic checklist approach [13]. All those studies used a fixed non-machine learning measures approaches that were not focused much on the bug reports.

This study aims to help software project managers, software engineers, security researchers, and testers [14] to evaluate and estimate the current existing risk of specific software components by analyzing historical bug records for specific attributes. This will be done through extracting bug-fix time prediction from the existing bug reports using a non-linear regression approach using Tensorflow tool [15] and analyzes duplicated bugs within these reports with their priority fixed levels. Therefore, builds a risk-driven estimation system based on the bug-fixed time, duplicate bugs, and bug priority level. This can be utilized for new requirements, design, and implementation phases in the already-built software cycle as insights that allow internal software stakeholders to identify what errors or bugs need to be addressed first and to help them to prioritize the mitigation process of the high likelihood and the highly impacted software vulnerabilities before and after the deployment phase.

The rest of the paper is organized as follows. Section 2 provides related work. Section 3 describes the methodology. Section 4 and Section 5 presents the experimental results and discussion. Finally, Section 6 and Section 7 represents the and limitations of the approach and the conclusion.

II. RELATED WORK

Software risk is the likelihood of a weak point to be exploited by threat agents. Despite the threat root cause, it is mandatory to consider the existing risk during the software project life cycle to prevent, prioritize, and mitigate the ones with a high negative impact on the software project. Several studies focused on the bug prediction fix-time approaches extracted from well-written bug reports. Kaur, G., and Bahl expressed that measuring software reliability is a difficult problem to obtain since there are many hardware and software attributes to consider [16]. This study found that achieving higher software reliability could be reached by using better development processes, risk management processes, and configuration management processes.

978-1-7281-9673-2/20/$31.00 ©2020 IEEE

Risk management has a high role importance on the software changes process. In E.hassan *et al.* study [3], they found that the accuracy of developers could reach a percentage of 96.1% of the time when they are experiencing bug-introducing changes. This study focused more on risky changes that can be effectively identified using factors such as the number of lines added by the change of the software, the availability of bugs in the software files, the number of changed bug reports, and the experience of the developer who is issuing the change. Byron J. *et al.* stresses the importance of implementing early software risk assessment and the need to categorize risk in software development projects [17]. This study stressed that risk categorization is the starting step for the goal of risk prioritization. To reach this objective the project manager should take into consideration different attributes such as project size, complexity, the risk of timing, and scheduling in the project life cycle. Also, this research addressed the role of the developer to be able to assess and understand risk factors through the software development life cycle (SDLC).

Well-written bug reports have a substantial role for project managers and developers to provide the project with the references needed for the software ongoing bugs issues. A study by Bettenburg N., *et al*. [18] explained the various attributes that affect bug report quality such as readability, content, categorization, and completeness of bug information. This study surveyed a group of developers with questions related to the resolution of the bug reports. The results showed that the developers got more attention to clear documented reports than the ones that are poorly written. Bug prediction modeling is also considered a new essential technique for the concerned project stakeholders. It is used to track the software development project and to assess the overall software quality through the different project phases. A study by Varuna G. *et al*. [6] showed different bug prediction models. The study used the identifications of bugs in the software project as a major indicator that affects the software's overall quality outcomes. This study showed that many different reasons for the bug to appear in the software project cycle such as depth of inheritance tree, weighted methods per class, and the coupling between objects. Moreover, this study concluded its prediction methodology results to enhance the performance of the overall project and to help to allocate sufficient software project resources.

Other bug prediction studies [19, 20, 21] focused on the importance of the prediction on the software maintenance phase and how it can help developers to prioritize bugs. These studies used five different predicting methods to estimate the bug-fix time prediction such as regression analysis, decision tree models, Markov model, Monte Carlo method, and kNN-based method. These studies reached a high rate prediction for all the five models used on their dataset and the study supported their scope with assumption probabilities for the number of bugs that can be fixed to build a converging prediction model. Predicting the time and effort to fix bugs is vital for project managers during the project life cycle. W. Abdelmoez used Naive Bayes network algorithms to predict the upcoming bugs time fix through analyzing the time for fixing a bug obtained from different classified bug historical reports [7]. Cathrin W. *et al.* predicted the effort for the bug fixes using the nearest neighbor approach and kNN methodologies [22]. The study found that the predicted effort value of the predicted bug-fix time could be used to generally allow better allocation of resources for project managers.

The mentioned studies with their different prediction approaches [19, 20, 21] stressed out indirectly the need to provide a well-written bug report as a machine learning datasets [18], where the outcomes to this are reflected on software reliability [16], software quality [1, 4], providing better sufficient project resources [6], and even estimate the effort needed for the developers to fix upcoming bugs [7, 22]. Furthermore, other studies focused on the importance of risk management [3], risk assessment [17] in change management processes and in software projects respectively. However, many risk estimation studies are focused on analyzing the external factors that affects the software to estimate risk levels [9, 10, 11, 12, 13]. Therfore, the current study is necessary to provide a quantitative and objective measure to calculate risk values based on the historical bug reports itself from the software components-level perspective with machine learning prediction approach. In this study we will focus on the approach of calculating a software risk estimation value from the available bug reports attributes that a risk value can be derived from through a certain dataset. This will show how it can be beneficial not only to the software development teams but also to organizations management and project managers to tackle threats as the software project continue to grow.

III. Research Methodology

In this section, the used dataset structure is discussed followed by a discussion about the proposed risk decision model used. Lastly, presented the evaluation criteria to evaluate the prediction results of the bug-fix time input using four regression metrics.

A. Used Dataset

This study is conducted on an online open-source bug repository *Mozilla Core* [23]. The definition of the dataset attributes and their data types are shown in Table 1. The most important attribute is the *Days_to_fix_the_bug*, where it represents the input points for the proposed prediction module. This attribute is derived from the difference between *Resolved_time* and *Created_time* attributes and converted into days' time format. These values are to be fed to the prediction module in advance from the dataset. In addition, the non-linear regression prediction algorithm has to predict only from the fixed bugs records, therefore only *Fixed* category values from *Resolution* attributes are considered. Table 2 lists the total number of the *Fixed* Resolution type, the total number of the *Duplicate* resolution type, the average bug fixing time, and the duration period for the selected *Mozilla Core* dataset.

TABLE I. THE STRUCTURE OF THE USED DATASET

Attributes	Definition	Type
Issue_id	The unique ID used to distinguish the bug.	Positive Integer
Priority	The priority number assigned to the bug.	String
Component	The component name the bug is located in.	String

Duplicated_issue	Contain duplicated bug ID. And used to identify if the same issue reported before.	Positive Integer
Title	Generic title to describe the bug.	String
Description	More details description for the bug.	String
Status	Assign by the project lead and has the following values: (Verified/Resolved/Closed)	String
Resolution	Specify whether the bug is (Fixed/ Invalid/will not be fixed/ already working or duplicated)	String
Version	To specify in which version or branch repository the bug is located.	String
Created_time	Date/time of the bug reported.	Date/Time
Resolved_time	Date/time of the bug resolved.	Date/Time
Days_to_fix_the_bug	The difference in days from the attributes *Resolved_time* and *Created_time*	Float

TABLE II. THE NUMBER OF FIXED BUG COUNT, DUPLICATED SAMPLES COUNT AND DURATION FROM THE DATASETS

Dataset name	**Number of the *Fixed* bug samples**	**Number of the duplicated bug samples**	**Average bug fixing time (days)**	**Duration of Bug Reporting Period**
Mozilla Core	101500	44692	419.59	Mar 1997 – Jun 1999

B. Risk Estimation Approach

Fig. 1 shows a simplification of the proposed risk decision process model. The proposed model takes two inputs, the component name of the bug in the project and the priority number that is assigned by the project manager to handle the urgency for the bug-fix and push it for the next pipeline tasks. Then, the two inputted data are filtered through the historical bug file reports as per the entered inputs. A prediction module is proposed and works on the filtered data to predict the bug fix time for the upcoming bug. After this, the value of predicted bug-fix time is handled to the risk analyzer module along with the earlier entered priority number. In addition, there must be also a consideration for the number of duplicated bugs related count to the same project component that is handled to the risk analyzer module. This is because the multi occurrences of the same bug mentioned on the bug report to the same software component could indicate a risk value level and must be considered within the proposed model. Finally, the risk analyzer module used all these attributes to find out the quantitative risk value.

The predicted bug fix time value is extracted through the use of the non-linear regression with neural network training [24]. Formula 1 shows the risk value obtained from the bug-fix time value.

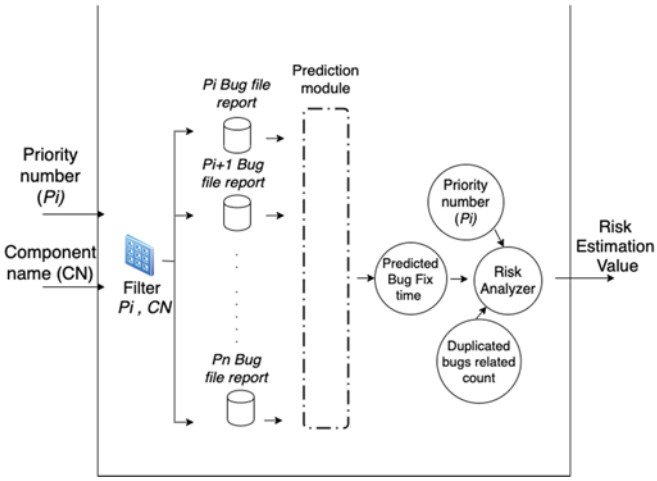


Fig. 1. The Proposed Risk Estimation Process Model

$$RB = \frac{L}{a} * 100\% \quad (1)$$

In Formula 1, L represents the value of the last predicted bug-fix time sample; a represents the average of the actual *Fixed* samples; *RB* represents the risk percentage of the bug-fix time prediction for the actual *Fixed* samples within the software component.

To calculate the risk values from the duplicated bug type category within the same software component, there will be the only consideration for the duplicated bugs that has a value of *WONTFIX* type since this category of bug records will not be fixed by the developer for some reason and might contain a risk level from not fixing it. Formula 2 shows the equation to calculate the risk from the duplicated bug records.

$$RD = \frac{DW}{T} * 100\% \quad (2)$$

In Formula 2, DW represents the total number of duplicated bugs with *WONTFIX* origin within the software component and including the count of the related *WONTFIX* records for each of the duplicated bug record; T represents the total number of duplicated bugs within the same component regardless of their relationship within bugs with the *WONTFIX* type; *RD* is the final risk percentage to the duplicated bugs issues.

The more assigned priority for the bug record issue listed in the bug report, the more final risk value it should have. Table 3 shows the risk percentage levels *RP* from each of the five priority levels obtained from the used dataset. Assuming *P1* to *P5* sorted based on highest to lowest priorities for the bug fix. These values will be considered and added up to the total risk level only if L does not equal a from Formula 1 or if $DW > 0$ from Formula 2. This condition is to make sure the risk value from priority types will only be considered if there exists a risk from the duplication bug issues or from the predicted bug fix time; otherwise, the value of the priority risk will be redundant.

The values of *RB*, *RD*, and *RP* are converted into a percentage of a maximum of 33.3% rate. Formula 3 represents the total final risk value (*TR*) calculated from each of the risk values mentioned earlier and to be within 100% of the total rate risk level.

TABLE III. THE RISK LEVELS ASSIGNED FOR EACH PRIORITY CATEGORY IN THE DATASET

Priority Type	Risk Priority level (RP)
P1	100%
P2	83.4%
P3	66.8%
P4	50.2%
P5	33.6%
Unassigned	16.6%

$$TR = RB + RD + RP \quad (3)$$

C. Evaluation Criteria

To evaluate the risk estimation approach, the bug-fix time prediction part needs to be evaluated as well. In this study, four regression metrics are used on the predicted bug-fix time. These metrics are mean absolute error (MAE), mean squared error (MSE), R^2 (R squared), and median absolute error (MedAE). These metrics measure the error rate between the actual and predicted bug-fix time through the ML algorithm. Assuming E is the actual bug-fix time from the dataset, $\acute{E}$ is the predicted bug-fix time, $\hat{E}$ is the mean of E, and n is the total number of the bug samples. The used metrics can be calculated as follow:

$$MAE = \frac{\sum_{i=1}^{n}|E_i - E'_i|}{n} \quad (4)$$

$$MSE = \frac{\sum_{i=1}^{n}(E_i - E'_i)^2}{n} \quad (5)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(E_i - E'_i)^2}{\sum_{i=1}^{n}\left(E_i - \hat{E}_i\right)^2} \quad (6)$$

$$MedAE(E, E') = median(|E_1 - E'_1|, \dots, |E_n - E'_n|) \quad (7)$$

IV. EXPERIMENTAL RESULTS

The experiment has been implemented on 2.2 GHz Intel Core i7 16 GB 1600 MHz DDR3 machine with 1536 MB GPU. The prediction of bug-fix time was conducted through Tensorflow [15] software library version 1.9.0.

The experimental results of one component from the Mozilla Core dataset called '*Networking: HTTP*' are shown in Table 4. To get the final risk value for each priority type, this study extracted the number of bug fixed samples for each priority category to calculate the actual bug-fix time average and using it with the last bug-fix time predicted value to calculate the *RB* value. Similarly, to calculate *RD*, the total number of duplicates bugs with the total number of duplicated bugs with *WONTFIX* origin is extracted. In addition, *RB* values are added based on the priority type assigned from Table 3.

V. DISCUSSION

As shown in Table 4, the total risk values ranged from 27.4% to 84%. The total risk values are strongly dependent and mostly proportional to risk values from the bug-fix time. *P2* and *P4* categories representing the topmost two risk values of *RB* with 158.7% and 135.4%, respectively to reach total risk value *TR* of 80.6% and 84% in order. *P4* category represents the best bug-fix time prediction accuracy of 0.35 R^2 value, where *P1* represents the worst prediction accuracy of 0.03 R^2 value. The highest *RD* is for the *P4* category with 66.6% for 3 duplicates bug samples, where the *Unassigned* category represents the second-highest for 59.2% for 635 duplicates bug samples.

Fig. 2 shows the graph representation for the bug-fix time prediction (represented in red marker) along with the distribution of the fixed samples (represented in blue markers) to get the values of *RB* for each component. Also, the values of the *last bug-fix time prediction* and *Avg fix time* are illustrated. The prediction accuracies used in the evaluation criteria are calculated from the bug-fix time prediction and which is also illustrated graphically. The accuracy rates of bug-fix time prediction with few *Fixed* samples count than *P1* represent the highest accuracies ranging from 0.17 to 0.35 R^2 value. Except for *Unassigned* and *P5* category *Fixed* samples, the prediction accuracy rates is propotional to the fewer *Fixed* samples count.

The bug-fix time prediction risk value (*RB)* and the priority level risk value (*RP*) represents the highest percentage-levels among the results. These two values, *RB* and *RP*, have been showing a strong effect on the final total risk values (*TR*). Also, due to its fewer number of duplicated samples within the same used software component, the duplicated bug risk value (*RD*) showed a minor effect on the *TR* values. *TR* was among the highest when both *RB* and *RP* were high as showing in the *P2* category reaching 80.6%. Similarly, in the *P1* category, when the value of *RB* is decreased, the *TR* also be decreased reaching 58.5%, even though the *RP* reached a high-risk percentage.

VI. THREATS TO VALIDITY

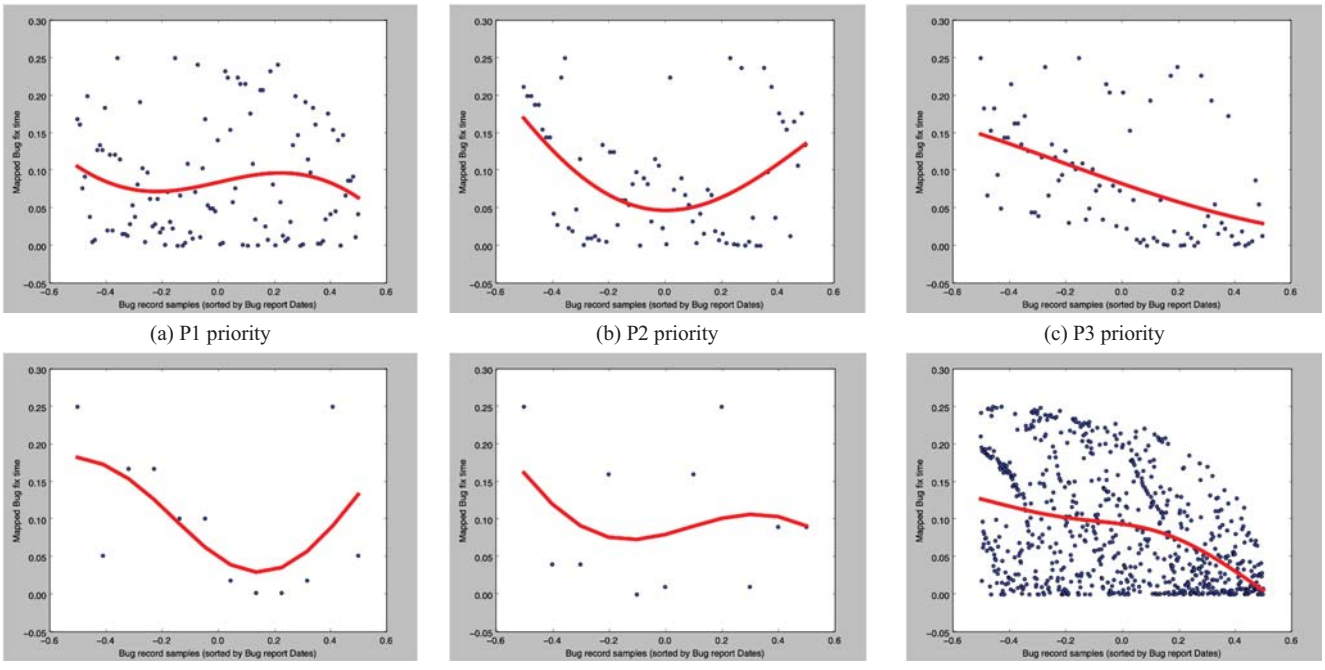
This study has similar bug-fix time prediction limitations that appeared by the approach used in [24]. Other technical factors, such as hardware issues and non-technical factors are not considered within the total risk values for the software component in the proposed approach. Additionally, the risk values results of this study are considered circumstantial to the dataset bug report attributes and do not reflect real risk value levels within the overall software project.

VII. CONCLUSION AND FUTURE WORK

Predetermining software risk levels for specific components are beneficial for internal software stockholders for managing their administrative and technical upcoming tasks. In this study, a proposed approach is implemented to estimate software risk levels on a specific software component from one bug report dataset providing risk values results from 27.4% to 84%. A similar approach will open for researchers to look for similar software risk level estimation and investigate different estimations techniques. The future work of this study will use natural language processing as a part of bug duplication identification for providing better-automated software risk level results.

TABLE IV. RISK RESULTS FOR 'NETWORKING: HTTP' COMPONENT CATEGORIZED BY PRIORITY TYPES

Priority types	No. of Fixed Samples	No. of duplicates releated to *WONTFIX*	No. of only *WONTFIX*	Total of duplicates samples	Last bug fix time prediction value (days)	Avg fix time value (days)	RD	RB	RP	TR	Evaluation criteria accuracy (Bug fix time prediction)
P1	113	0	0	10	0.064	0.084	0%	75.6%	100 %	58.5%	MAE: 0.0638 MSE: 0.0055 R^2: 0.0390 MedAE: 0.0619
P2	76	0	0	11	0.135	0.085	0%	158.7%	83.4%	80.6%	MAE: 0.0545 MSE: 0.0045 R^2: 0.2247 MedAE: 0.0492
P3	84	8	11	50	0.029	0.085	38.0%	34.9%	66.8 %	46.5%	MAE: 0.0514 MSE: 0.0045 R^2: 0.2231 MedAE: 0.0420
P4	12	1	1	3	0.133	0.098	66.6%	135.4%	50.2%	84.0%	MAE: 0.0542 MSE: 0.0049 R^2: 0.3594 MedAE: 0.0386
P5	11	1	1	4	0.091	0.1	50.0%	91.3%	33.6%	58.2%	MAE: 0.0706 MSE: 0.0064 R^2: 0.1777 MedAE: 0.0734
Unassigned	732	135	241	635	0.005	0.83	59.2%	6.6%	16.6%	27.4%	MAE: 0.0573 MSE: 0.0047 R^2: 0.1667 MedAE: 0.0533



(a) P1 priority (b) P2 priority (c) P3 priority

(d) P4 priority (e) P5 priority (f) Unassigned priority

Fig. 2. Bug-fix time prediction results for 'Networking: HTTP' component categorized by priority types.

REFERENCES

[1] M. Alenezi, S. Banitaan, and Q. Obeidat. Fault-proneness of open source systems: An empirical analysis. *Synapse*, 1:256, 2014.

[2] M. Abdulshaheed, M. Hammad, A. Alqaddoumi, and Q. Obeidat. Mining historical software testing outcomes to predict future results. *Compusoft*, 8(12):3525–3529, 2019.

[3] E. Shihab, A. E. Hassan, B. Adams, and Z. M. Jiang, "An industrial study on the risk of software changes," in *Proceedings of the ACMSIGSOFT 20th International Symposium on the Foundations of Software Engineering*.

[4] Barbara Kitchenham and Shari Lawrence Pfleeger. Software quality: the elusive target[special issues section]. *IEEE software*, 13(1):12–21, 1996 lll

[5] Gary McGraw. Software assurance for security. *Computer*, 32(4):103–105, 1999.

[6] V. Gupta, N. Ganeshan, and T. K. Singhal, "Developing software bug prediction models using various software metrics as the bug indicators," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 6, no. 2, 2015.

[7] W. Abdelmoez, M. Kholief, and F. M. Elsalmy, "Bug fix-time prediction model using na¨ıve bayes classifier," in *2012 22nd International Conference on Computer Theory and Applications (ICCTA)*. IEEE, 2012, pp.167–172.

[8] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller, "How long will it take to fix this bug?" *in Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*. IEEE, 2007,pp. 1–1.

[9] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. Characterization and prediction of issue-related risks in software projects. In *2015 IEEE/ACM12th Working Conference on Mining Software Repositories*, pages 280–291. IEEE, 2015.

[10] Andrea Herrmann and Barbara Paech. Practical challenges of requirements prioritization based on risk estimation. *Empirical Software Engineering*, 14(6):644–684, 2009.

[11] Christian Haisjackl, Michael Felderer, and Ruth Breu. Riscal–a risk estimation tool for software engineering purposes. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, pages 292–299. IEEE, 2013.

[12] Robert J Walker, Reid Holmes, Ian Hedgeland, Puneet Kapur, and Andrew Smith. Alightweight approach to technical risk estimation via probabilistic impact analysis. In *Proceedings of the 2006 international workshop on Mining software repositories*, pages 98–104, 2006

[13] Chandan Kumar and Dilip Kumar Yadav. A probabilistic software risk assessment andestimation model for software projects. *Procedia Computer Science*, 54:353–361, 2015.

[14] N. Alsolami, Q. Obeidat, and M. Alenezi. Empirical analysis of object-oriented software test suite evolution. *International Journal of Advanced Computer Science and Applications*, 10(11), 2019.

[15] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin,S. Ghemawat, G. Irving, M. Isard,et al., "Tensorflow: A system for large-scale machine learning," in *12th{USENIX}Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp.265–283.

[16] G. Kaur and K. Bahl, "Software reliability, metrics, reliability improvement using agile process," *International Journal of Innovative Science, Engineering & Technology*, vol. 1, no. 3, pp. 143–147, 2014.

[17] Byron J Williams, Jeffrey Carver, and Rayford B Vaughn. Change risk assessment: Understanding risks involved in changing software requirements. In *Software Engineering Research and Practice*, pages 966–971, 2006.

[18] N. Bettenburg, S. Just, A. Schr¨oter, C. Weiss, R. Premraj, and T. Zim-mermann, "What makes a good bug report?" in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. ACM, 2008, pp. 308–318.

[19] P. Bhattacharya and I. Neamtiu, "Bug-fix time prediction models: can we do better?" in *Proceedings of the 8th Working Conference on Mining Software Repositories*. ACM, 2011, pp. 207–210.

[20] E. Giger, M. Pinzger, and H. Gall, "Predicting the fix time of bugs," in *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*. ACM, 2010, pp. 52–56.

[21] H. Zhang, L. Gong, and S. Versteeg, "Predicting bug-fixing time: an empirical study of commercial software projects," in *Proceedings of the 2013 international conference on software engineering*. IEEE Press, 2013, pp. 1042–1051.

[22] C. Weiß, R. Premraj, T. Zimmermann, and A. Zeller, "Predicting effort to fix software bugs," in *Proceedings of the 9th Workshop Software Reengineering*, 2007.

[23] Zhu, Jamie. BugRepo. Jul. 2018. Accessed on: Nov. 1, 2019. [Online]. Available: https://github.com/logpai/bugrepo

[24] Hussain Mahfoodh and Mustafa Hammad. Bug-fix time prediction using non-linear regression through neural network. In *Proceedings of the 3rd Smart Cities Symposium*, 2020.