Automated Software Bug Management System for Small-Scale Organization

SMKH Hemali# and TGI Udayangi

General Sir John Kotelawala Defence University, Sri Lanka

#35-it-0046@kdu.ac.lk

Abstract -During software running, even errors due to system complexity and inadequate testing may occur. Troubleshooting plays an important role in software development and evaluation steps. Due to rapid changing technology, the whole system should adapt according to the situation, including matters such as well-skilful persons, technology and data. The bug management process has several steps, and controlling those steps is a huge challenge. Behind the situation, the small-scale Software companies need resources other than local organizations. This research focuses on identifying local small-scale organization behaviours, since they have fewer financial problems and less technological literacy of operating some licensed automation tools used in the software bug management industry. The research raises how automation techniques solve financial challenges faced by small-scale organizations. A research methodology approach which analyses previous studies and collected data is observed, and that information is validated according to the small-scale organization requirement. Finally, a proposed a system to overcome the situation is introduced, which is a web-based application that hosts the cloud. The proposed system implementation provides a facility for real-time communication between SQA, developers, and other team members via comments on each reported bug, while it assigns bugs to all the job roles represented by the agile software development life cycle, to reach historical bug records. Facility to embed the technical evidence as a report to the bug for a better understanding of the developer is also introduced. This facility generates reports for tracking each developer and tester's performance of that particular local organization. The proposed system uses an open-source development framework.

***Keywords**: bugs, automation, management, financial, developer, literacy, small-scale*

I. INTRODUCTION

During every stage of a given software development process; the corresponding software system generates miscellaneous types of defects. Then before developing a bug management application requires a deep understanding of software bug characteristics. But this process's initial step is estimating the bug. An efficient bug management process is critical for the success of software projects. Bug case to decrease the software reliability, quality, security, and vital area. Identifying and tracking these defects efficiently has a measurable impact on software reliability. (Strate and Laplante, 2013),(Jalbert and Weimer, 2008). The bug management process is a very difficult task, and it has prior work to do. For example, by automating bug triaging, detecting duplicate bugs, and understanding the rationale for re-opening bugs. The process of managing bugs involved several human resources such as developers, testers, reporters, project managers, product owners. (Ohira *et al.*, 2012)., Some of the bugs that developers encounter while working are having to keep separate records and update them These often have to be done manually. It takes extra time because there is a separate process to manage a bug and it consists of several steps.(Mujtaba, Mahmood and Nasir, 2011).Bug management can become even more challenging when the development projects are large. Moreover, bad fixes may cause the injection of new defects in the software. Even small-scale organizations faced related same situations within the industry. (Mujtaba, Mahmood and Nasir, 2011)

The research objective is to identify if a small-scale organization have sufficient budget to handle previous process and problems while encountering the defect managing. Because most of the time small-scale organizations used open-source tools to manage bugs. Bugzilla and the ITracker are leading open-source tools in the industry. But JIRA has licensed software used to manage bugs. Bugzilla is depending on platforms, and it has a lower ability to customize. It is dependent on the SQL database. Then highly required opensource tool without those limitations. Another objective is to identify if they face any problems without having a customizable application. (Serrano and Ciordia, 2005),(Grottke and Trivedi, 2007). The goal of this research is to introduce a new system to overcome such a situation. The proposed system should be purchased for a low price. It is platform-independent and highly customizable.

II. RELATED WORK

A. *Automated duplicate detection for bug tracking*

Bug tracking is one of the stages in the software bug management process, that process can identify duplicate bugs. And manual bug identification was a time-consuming process of the current software development industries, it adds a higher extra cost for the development process. That system used surface features, textual semantics, and graph clustering to the prediction for the duplicate bug report. The author used 29,000 datasets from using the Mozilla project. This system should be able to develop costs by filtering 8% of duplicate bug reports. the textual analysis they used an algorithm to calculate the similarity of the document. (Jalbert and Weimer, 2008) Software bug management systems store valuable data for testing hypotheses concerning maintenance, building statistical prediction models, and investigating developer effectiveness. For the latter, issue tracking systems can be mined to explore developers' emotions, sentiments, and politeness. But detection in software artifacts is still in its early stage due to the lack of manually validated data and tools.

B. *Emotional Side of Software Developers in Jira*

The working environment is always not satisfactory. Software developers also face this situation. (Ortu et al., 2016) provided a label of emotions present on issue comments. This paper authors manually labelled 2,000 issue comments and 4,000 sentences written by developers with emotions such as love, joy, surprise, anger, sadness, and fear. An efficient bug management process is critical for the success of software projects.

C. *Impact of bug management patterns on bug fixing* The author of this paper has been mentioned it has prior work focus on improving automating bug triaging, detecting duplicate bugs, and understanding the rationale for re-opening bugs. This paper introduced four patterns and the different relations between the people involved in the process: reporter, developer, tester of a bug. Their case study is based on Eclipse Platform and Java Development Tool (JDT) projects. Presenters of this paper were demonstrated that these patterns have an impact on the efficiency of the bug management process. Their conclusion was to improve their efficiency through better communication about bugs before assigning them. (Ohira et al., 2012)

D. *Bugzilla, ITracker, and other bug trackers*

Bugzilla and ITracker are the existing open-source software of the bug management industry used. Bugzilla was facilitating to input new bugs or search for, track, or edit existing ones. It had two methods to track bugs when you submit the bug to that system your mandatory input product, component, version keywords, severity, attachments, and dependencies fields are related to fixing the bug. Another way is system-generated reports. Bugzilla is a web-based application. ITracker is an issue-tracking system designed by Jason Carroll in 2002. It was supported multiple projects with independent users. Its features resemble Bugzilla's. When comparing those two systems main difference is ITracker is platform-independent and database-independent. (Serrano and Ciordia, 2005).

E. *Bug Characteristic of Open-Source Software*

Manually priorities to bugs were resource-consuming. Researchers were mentioned single feature is used which leads to information loss because bugs have a lot of features including "severity", "component'', "status", "assigned to", "summary" etc. But this paper introduced the solution as an improved model based on problem title, severity, and component for bug prioritization. They used term frequency and inverse document frequency to convert textual features to numeric features. They used a special algorithm to overcome the complexity of such feature-generated data. The algorithms are non-negative matrix factorization, principal component analysis. In this research on average maximum accuracy is achieved by SVM with Non-negative Matrix Factorization (NMF) and X-mean clustering. (Iqbal et al., 2020). When developing effective tools for bug management want a deep understanding of the software bug characteristics. This research paper was based on open-source projects running by the Linux kernel, Mozilla, and Apache. They collected a sampling of 2,060 real-world bugs. The manual study is separated into three dimensions like root causes, impacts, and components. suggesting more support to help developers diagnose and fix security bugs, especially semantic security bugs. (Tan et al., 2014).

F. *How to Chat Technology Enables Social Translucence in Bug Report Activities*

Software bug management is a daily work routine in software engineering. The paper may focus on the use of chat technology in software engineering by analysing the coordination between client and vendor in a large government software project in Brazil (Gov-IT). author of this paper was used two methods collected data for their work live and online interviews. They used chat technology to coordinate their cooperative work by enabling the participants to monitor the availability of developers and the urgency of detecting bugs synchronously. According to their conclusion, understand the contextual nature surrounding bugs faster than using the bug tracking system. (Tenório, Pinto and Bjørn, 2018).

G. *Machine Learning Techniques for Software Defect Detection*

Machine Learning approaches are a trend of problem-solving. Machine learning is a vast area used in the software development industry. Machine learning techniques are proven to be useful in terms of software bug prediction. The paper is used to analysed to public data set of software modules and provides comparative performance analysis of different machine learning techniques. Software companies are spread the world widely. Then when developing the software, the quality problem is a leading issue for the software industry. The industry is suffering and closing for this issue. In this circumstance, it is important to call and remove its root cause. Recently industry economic loss will increase. (Aleem, Capretz, and Ahmed, 2015)

H. Challenges of Software Quality Assurance and Testing

Hossain trying to show some vital challenges of software quality assurance and testing which have been facing by software industries. This research covered both local and international organizations. that paper introduces the different categories of challenges along with responsible stakeholders. And they search and experiment the testing tools are available testing elements are available testing process has improved but still software has some testing challenges. The conclusion of the author is switching the systematic approach to solve the problem. (Hossain, 2018)

III. METHODOLOGY

In this research, a most typical strategy is to collect data from various sources for study and refer to past analysis and analyses of previous research publications and sort some key information. According to this research used questionaries for gathered information. This research gathered information from existing problems within small-scale IT organizations. This research identified information by giving questionaries to all designation such as Quality Assurance (QA), Developer, BA, etc. Use of small IT base existing companies. The statistical sample was included "Enuri information system (Pvt)"," Alpha information system (Pvt)" etc. After gathering 300 sample data, analysis of them to identify research problems. It identified the main key problems that breach from each role. Identifiers were mapped with the related work. This research recognizes mapping issues raised by earlier researchers; company issues will be tailored to the needs of the organization to continue the process.

IV. ANALYSIS

After analysis of previous problems, it was a clear need for the new system to overcome such problems. The small-scale organization had to struggle to reduce the budget and compatible the complex functions. The proposed system is specially developed by a selected small-scale organization. The proposed system maintains easy icons and user-friendly interfaces to identify the functions. It was designed to facilitate

Prioritization of the Bug, reviewed the progression of the developers, analyze the developer's quality by tracking the number of bugs, generate a monthly bug managing report, provide a facility to communicate between testers, developers, and other team members via comments on each reported bug, provide facility to reviewed bug history records. predict each developer's performance, provide the facility to embed the technical evidence (automation test technology-based report (testNG) report) as a report to the bug for a better understanding of the developer.

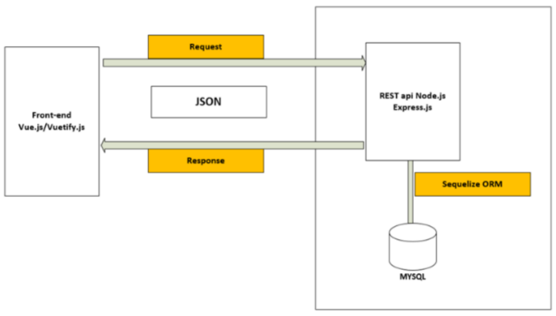


Figure 1.Communication process between front end and back-end
Source: Author(s)2021

The previous diagram explains how to communicate the system between the front-end and back-end using JSON. JSON is a lightweight format for storing and transporting data. JSON has an attribute "self-describing data" and "easy to understand data". JSON is the most suitable technology for transport data that generate Vue.js and Vuetify.js

V. DESIGN

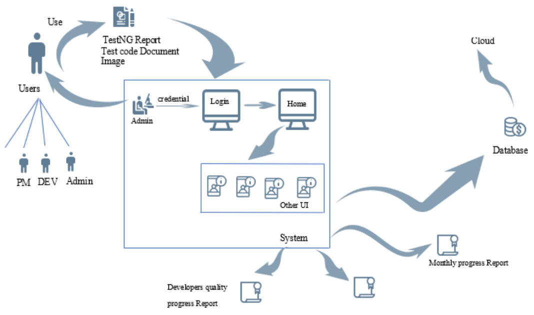
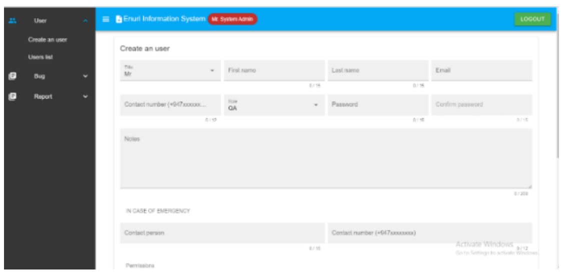


Figure 2.Conceptual design of the system
Source: Author(s)2021

According to the overall system architecture, four types of user roles could create by system administrator. And the system inputs are testNG reports, test code documents, test code images. Initially, the admin login to the system, and after he/she will create the user role as a software tester, developer, project manager, product owner. Then each user role will enter their credential and log in to the system. It will provide the ability to functioning the system. The system output is system-generated reports. The system-generated reports are developer progress, bug progress, monthly progress report.

This system is a web-based system, and it was hosted by the cloud. This proposed system has 5 modules as Bug, Report, User, label, project. Each module has divided into sub-modules. Every Module manages a unique task. In the Bug module, the User can create bugs, view bug history, edit bugs. But only the Software Quality assurance engineer has permission to end bugs. Only permitted Project module for the project manager, chief operating officer, Product owner. This module provides the facility to create a new project and see current project details, edit the previously assigned project details. And Label module provides the facility to create a new label for newly created bugs or it can be edit previously once. User modules provide all the administration processes. This module provides a facility to create user roles and assign user permission. Only system admin can create user roles. The report module provides system generate reports.

VI. IMPLEMENTATION



Title: Mr
First name
Last name
Email
Contact number (+947xxxxxx...
Role: QA
Password
Confirm password
Notes
IN CASE OF EMERGENCY
Contact person
Contact number (+947xxxxxxx)
Permissions

Figure 3. Web admin interface of the system
Source: Author(s)2021

This project is an online web-based system. And this system will be doing to overcome previous research problems. The system may have only one login page. Because only the system admin creates the user roles and after created the user role, that user can be logged into the system using the credential. But the proposed system restricts some user permission from those who are not relevant to each job role. For example, only Software Quality Assurance (SQA) has permission to delete the bugs inside the system. SQA assigns the bug to the system, it will generate a separate bug id for each unique bug. And had permission to edit the previously assigned bugs for every job role that is traded. Only project managers can assign the project to the proposed system. The system implements using node.js, vue.js, and express.js technologies. JSON is the transmitted media of data from front end to back-end JavaScript, in the back-end development technology.

The system will display the status of the bug such as in queue, in due, hold, completed, QA pass. The proposed system generates a monthly bug progress report. This report will include a count of the fixing bugs in each project, the number of bugs that fail to fix, total bugs assign to the system. The system data will be host in the cloud database. That takes as a data backup. When considering the system's non-functional requirement, security act a major role. To increase the security of the system, only admin and SQA have been permitted to remove assigning bugs to each developer. The proposed system data was stored by the "MySQL" database. The system was implemented to submit the bug as image, screenshot, and embedded technical report such as (testNG report) using the Eclipse framework.

The proposed system can manage several types of bugs such as system performance bugs, functional bugs, non-functional bugs, Security, Compatibility, Usability bugs. When SQA assigns bugs mentioning the severity of the bug, then the proposed system filters bugs according to their severity. Based on the severity of bugs can be divided into critical, high-severity, medium-severity, and low severity. While assigning bugs, SQA uses a process of prioritization to separate each bug. Based on priority, it can be divided into urgent, high-priority, medium-priority, low-priority. The proposed system provides a facility to manage bugs of this type. This proposed bug managing system can be used by any local small-scale software development organization. Because most of all the software development companies follow the same life cycle. They are the main stakeholders of this product. The proposed system is most appropriate for project teams who have fully automated test suites for their testing process because proposed system provide embedded facility to attach automated test report as technical evidence. Each stakeholder can connect without any conflict after hosting it in the cloud, the organization can manage system-generated reports via cloud providers.

After implementation of the proposed system, it can be generated different types of system generated reports, that report evaluates the whole system. The report includes total bug count input to the system, solved bug count, exist, bug count, an average of solved bug count, and measure Developer's performance count of bugs they solve within a period. And measure the Software quality engineer's performance also. The report shows the final summary of the month and Year. Then it is used to calculate the efficiency and effectiveness of the employees and the organization. System-generated reports generate Quarterly, monthly, or weekly. If anyone needs to customize the report then the system provides that facility also. Finally in the evaluation part system run a fully functional testing round to ensure the system functions run without any blockers.

VII. LIMITATION

The function of the system is always efficient when it has an automated test code of its application. There is no module for tracking real-time bugs.

While configuring some software it may generate compatibility issues, and always struggling those issues.

When used the SQL database, it is spending extra time to configure and write some quarry.

VIII. FUTURE WORKS

Bug reporting steps are an uncoordinated distributed process. Therefore, many duplicate reports are being generated. to address this issue, there is a need for an automated duplicate report detection approach. In the fracture, researchers will plan to create software to track this problem and to extend this study by investigating more bug reports from different software systems. Researchers also would like to search to improve the accuracy of duplicates.

In facture enhancement, this approach could explore make fully automated, that means when bug tracking section module is added. The other enhancement is the wish to connect the eclipse framework to that system to upload the automated test report.

Researchers will enhance that bug report security using classification and machine learning algorithms. There are different types of supervised learning and un supervise learning algorithms are available.

The system will enhance connecting social media platforms, then employees could improve accountability of communication activities related to bug management without large effort.

IX. CONCLUSION

The Software Development industry is constantly looking for new ways to implement services and always decrease time consumption and cost of managing problems, while at the same time they are struggling with some incidents. There is a clear need for an automation bug management process that combines previous separate activities. The number of bugs or defects can cause significant financial losses for both software developers and customers. There may be a high probability when small-scale organizations haven't a sufficient budget to handle that situation. The proposed system has been provided to the open-source platform-independent application of highly customizable. That application provides a less complex function, and it has a user-friendly interface. It is portable. The proposed system facilitates the common communication platform. It has been providing a facility to import technical reports as evidence of a bug.

REFERENCES

Aleem, S., Capretz, L. F. and Ahmed, F. (2015) 'Benchmarking Machine Learning Techniques for Software Defect Detection', *International Journal of Software Engineering & Applications*, 6(3), pp. 11–23. doi: 10.5121/ijsea.2015.6302.

Grottke, M. and Trivedi, K. S. (2007) 'Fighting bugs: Remove, retry, replicate, and rejuvenate', *Computer*, 40(2), pp. 107–109. doi: 10.1109/MC.2007.55.

Hossain, M. S. (2018) 'Challenges of Software Quality Assurance and Testing', *International Journal of Software Engineering and Computer Systems*, 4(1), pp. 133–144. doi: 10.15282/ijsecs.4.1.2018.11.0044.

Iqbal, S. *et al.* (2020) 'Determining Bug Prioritization Using Feature Reduction and Clustering with Classification', *IEEE Access*, 8(August 2009), pp. 215661–215678. doi: 10.1109/ACCESS.2020.3035063.

Jalbert, N. and Weimer, W. (2008) 'Automated duplicate detection for bug tracking systems', *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 52–61. doi: 10.1109/DSN.2008.4630070.

Mujtaba, G., Mahmood, T. and Nasir, Z. (2011) 'A holistic approach to software defect analysis and management', *Australian Journal of Basic and Applied Sciences*, 5(6), pp. 1632–1640.

Ohira, M. *et al.* (2012) 'The impact of bug management patterns on bug fixing: A case study of Eclipse projects', *IEEE International Conference on Software Maintenance, ICSM*, pp. 264–273. doi: 10.1109/ICSM.2012.6405281.

Ortu, M. *et al.* (2016) 'Expectations, outcomes, and challenges of modern code review', *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016*, pp. 480–483. doi: 10.1145/2901739.2903505.

Serrano, N. and Ciordia, I. (2005) 'Bugzilla, ITracker, and other bug trackers', *IEEE Software*, 22(2), pp. 11–13. doi: 10.1109/MS.2005.32.

Shaffiei, Z. A., Mokhsin, M. and Hamidi, S. R. (2010) 'Change and Bug Tracking System: Anjung Penchala Sdn. Bhd.', *International Journal of Computer Applications*, 10(3), pp. 28–34. doi: 10.5120/1460-1974.

Strate, J. D. and Laplante, P. A. (2013) ‘A literature review of research in software defect reporting’, *IEEE Transactions on Reliability*, 62(2), pp. 444–454. doi: 10.1109/TR.2013.2259204.

Tan, L. *et al.* (2014) ‘Bug characteristics in open source software’, *Empirical Software Engineering*, 19(6), pp. 1665–1705. doi: 10.1007/s10664-013-9258-8.

Tenório, N., Pinto, D. and Bjørn, P. (2018) ‘Accountability in Brazilian Governmental Software Project: How Chat Technology enables Social Translucence in Bug Report Activities’, *ECSCW 2018 - Proceedings of the 16th European Conference on Computer Supported Cooperative Work*, pp. 1–27.

ABBREVIATIONS AND SPECIFIC SYMBOLS

[testNG] automation test technology-based report

[SQA] Software quality assurance

[NMF] Non-negative Matrix Factorization

[PCA] Principal Component Analysis

[MYSQL] Relational database management system

[QA] Quality assurance

[COVID-19] Coronavirus disease

[Gov-IT] Government Information technology

[JDT] Java development tool

[Bugzilla] software quality assurance application

[ITracker] Open-source bug management software

[JIRA] Software Management application

[JSON] Stands for JavaScript object notation.

AUTHOR BIOGRAPHIES

SMKH Hemali currently a 4th-year student in Information Technology department at General Sir John Kotelawala Defence University.

TGI Udayangi (B.Sc. Hons in information Technology (UOM), master’s in business administration in Information Technology (UOM)is a lecture (Probationary) in General Sir John Kotelawala Defence University. She has years of Experience as Quality Engineering at DirectFN and Pearson.