International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-5, March 2019

Mapping Bug Reports to Relevant Developers

Ayesha Khan, Gudapati Syam Prasad

Abstract**: *Every time a bug reportis newly generated, developers are always required to recreate the bug, execute code and performa nalysis to know the cause, this process is laborious and time absorbing. A tool which maps the bug reports to their relevant developers, preserving how the bug is solved and analyzing how many bugs are solved till date can deprecate the time absorbing to analyze bugs. This paper introduces an adjustable mapping approach where bugs with triage can be assigned to the developers who have experience in solving that particular type of bug and once the bug is solved the solution, status of the bug can be preserved in the bug repository. Based on the preserved values, we use the instance selection approach to find the number of bugs solved under a particular time period using time stamp and analyze the intensity of the bugs in the form of employee analysis and performance analysis. We also introduce community support where the developers can raise the questions with respect to bugs.***

Index Terms**: *Mapping Bug Reports, Bug triage, Bug Repository, Timestamp, Bug Analysis.***

I. INTRODUCTION

A bug in software is a mistake in the code which causes unexpected performance of the software component.[1]In a software team, both managers and developers extensively use bug reports in their routine development cycle.[2]Developers always have to recreate the bug and execute whole code to know the cause. Software companies deal with a huge number of bugs day to day[3]Mostly, the money of software companies is involved in fixing bugs. When a bug occurs, a developer will report the bug report in the form of a bug document. The number of bug reports that arise in the development life-cycle of a software component could be huge. Bug repositories are maintained for large projects that collect all information related to bugs. In bug repository, A bug report is made up of textual information related to the bug which can help in fixing a bug. It's a challenging task for the management to categorize all the bug reports they received. [4] A mechanical bug triage approach is proposed to avoid the huge cost of physical bug triage. The text classification techniques are applied to anticipate developers for bug reports. In this approach, a document gets mapped with a bug report. [5] A classifier predicts a list of developers, then they were ranked according to their priorities in Developer Prioritization. Thus, the developers with similar probabilities were discriminated using the Developer Prioritization. [6]

Manuscript published on 30 March 2019.
***Correspondence Author(s)**
Ayesha Khan, Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, A.P, India.
Dr. Gudapati Syam Prasad, Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, A.P, India.

Table. 1. Literature Survey

Paper	Method	Conclusion
"Who should fix the bug?" [5]	A mechanical bug triage approach is proposed to avoid the huge cost of manual bug triage.	The bugs are assigned by a human based on his/her expertise.
The Developer prioritization in bug repositories [6]	A classifier predicts a list of developers, then they were ranked according to their priorities in Developer Prioritization.	Developers aren't assigned bugs based on their tasks which leads to no improvement in bugs.
Bug Triaging: Profile Oriented Developer Recommendation [7]	A profile is created for every developer in Profile Oriented Developer Recommendation based on his/her previous work.	The corresponding area of expertise of each developer is indicated by mapping this profile to a domain mapping matrix (DMM).
Towards the graphical models for the text processing," Knowledge Information Systems [9]	A theory of distance graphs is suggested for the text representation and processing.	Similarity search isn't studied in this approach.

While a profile is created for every developer in Profile Oriented Developer Recommendation based on his/her previous work. Then, the corresponding area of expertise of each developer is indicated by mapping this profile to a domain mapping matrix. [7] To bypass low-factor bug reports in bug triage, unlabeled bug reports were combined with labeled ones to train a semi-supervised classifier so that the accuracy of the classification improves along both the labeled and unlabeled bug reports. A weighted recommendation list (WRL) is developed to boost the bug triage effectivity on unlabeled bug reports. This WRL predicts a few developers to whom we can assign an unlabeled bug report. [8] A theory of distance graphs is proposed for text representation and processing. The distances between distinct words in a document are represented as distance graphs. The information about the relative employment of words with respect to each other was maintained by distance graph representation. [9]
Three open source projects were designed to conduct surveys between developers and users. How to make a good bug report and what can be done to advance the nature of the bug is defined based on these survey results. [10] In this paper, we introduced a tool where a bug can be assigned to a developer who has experience in solving that particular type of bug. For example, a bug related to JAVA program will be assigned to a JAVA developer vice versa. Once the bug is solved, the status and the solution of the bug report will be saved in the bug repository which helps us to analyze the intensity of the bugs. When a bug which has recently solved occurs again then with the help of instance selection approach we can easily find out how the bug was solved in the previous period of time. This saves a lot of time, labor and manual work. The paper is formulated in the following manner:

*Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

Introduction & Literature survey in Section 1,Experimental Implementation, and prototype in section 2,Results and Discussion in section 3, and the Conclusion in section 4.

II. IMPLEMENTATION

The scope of the proposed work is as follows:

1. Assigning a bug to the respective developer based on his/her expertise.
2. Preserving the solution and status of the bug in the bug repository once it is solved by the developer.
3. Checking the bug status and intensity of the bugs.
4. Checking the bugs occurred under a particular instance for future purpose.

Fig. 1 illustrates the basic architecture of "Mapping Bug Reports to relevant developers".
A manager assigns bugs to different developers based on their expertise. The developers can communicate with each other by means of community support similar to a blog.

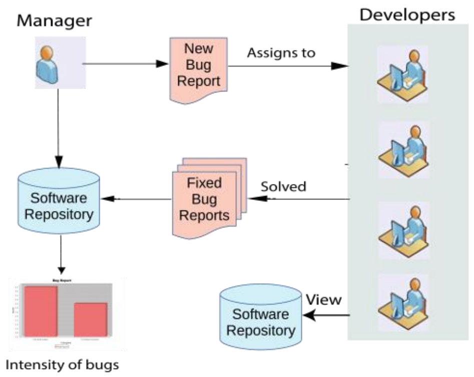


Fig. 1. System Architecture

After a bug is solved, the bug status and solution are saved along with the date in the bug repository from which we can make the instance selection. With the help of the data stored in the software repository, we can find out the intensity of the bugs. This system is implemented in JAVA language which is platform independent. A manager can add any number of bugs as there is no restriction. We use MySQL Workbench to store all the data. Since we developed the community blog, the developers can easily solve their bugs by questioning other developers and answering other developers.

A. MANAGER MODULE

Manager tasks are as follows

a. Assigning Bugs
b. Checking Bug status
c. Reviewing Bug Repository
d. Checking Timestamp
e. Checking Bug Intensity

*a. Assigning Bugs:*Fig. 2 depicts the data which a manager has to input while assigning a bug to a particular developer. *Bug Name* and *Bug Description* depends on the bug report. In *Assign To*, the manager will choose the developer to whom we want to assign a particular bug based on his/her expertise. For example, Karthik-Java Developer, Hamza-C developer. Bug Triage illustrates 3 levels of the bug namely: low, mid, severe. A date can be selected with the help of *datepicker.* Bug status will be selected as *Raise Token* to assign to a developer.
Once the bug is raised, the bug report with the developer to whom it is assigned is saved in the repository as shown in

Table. 2: Assigned bugs Table

Bug Name	Assigned To	status
on page	Aswin	Unresolved
Javac not found	Karthik	Unresolved
Application Crashes	Saraswathi	Unresolved

Table 2. Until the bug is resolved by the developer it shows the status as "UNRESOLVED".

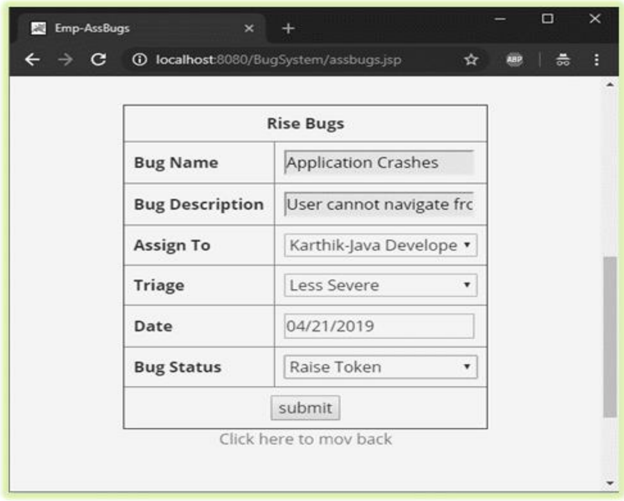


Fig. 2. Assigning Bugs

*b. Checking Bug status:*Fig. 3 depicts the status of a bug if is RESOLVED or UNRESOLVED, bug priority (bug triage) and to whom a particular bug is assigned. Once a developer solves a bug, the status of the bug will be changed to RESOLVED, else its status is UNRESOLVED.

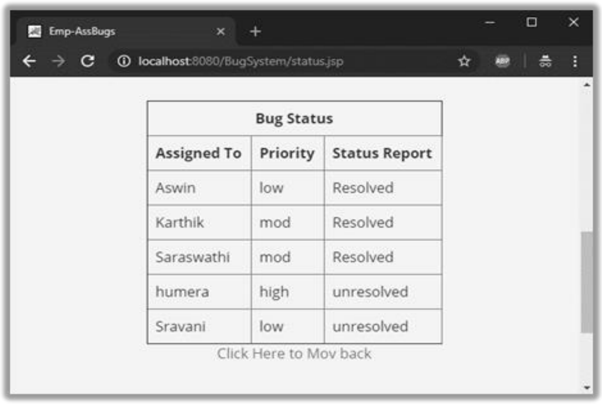


Fig. 3. Checking Bug status

*c. Reviewing Bug Repository:*For Example, Table. 3 depicts the information about Bug Name, Bug Description, Bug Triage, and its solution. This helps us to save the solution of the bug which is solved. If the bug isn't solved then it shows the solution as "WAITING" which mean the bug is yet to be solved. All the information is saved in MySQL Workbench and it helps in solving future bugs.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-5, March 2019

Table. 3: Bug Repository

Error	*Description*	*Triage*	*Solution*
on page	on page	low	dont know
Javac not found	cant compile programs	mod	set path for javac
Application Crashes	User cannot navigate from one page to another page	mod	The query is updated
Navigation error	On tapping signup button application crashes	high	waiting
Cannot upload profile picture	The database doesnt save the picture	low	waiting
Error in the size of the display picture	Pixels should be changed	low	pixels are changed
The application gets disconnected	Program error	low	There was an error in the code and it is solved

d. Checking Timestamp: Fig. 4 depicts the Timestamp i.e., the manager should select a particular instance of time in the terms of Start Date and End Date to see how many bugs are raised under a particular time period. And Fig. 12 depicts the Instance Selection of the bugs that are raised under the particular timestamp selected in Fig. 4. From Algorithm 1, it is clear that SD is the start date and ED is the end date. A timestamp is selected and the data is selected from the repository between those dates and a table is generated from the selected dates.

Algorithm. 1: Instance Selection
Step 1: Select a Start Date, SD.
Step 2: Select an End Date, ED.
Step 3: While (between SD && ED)
Select bug name, bug description, Triage, Developer, date from rep;
Step 5: Generate Table;
Step 6: End

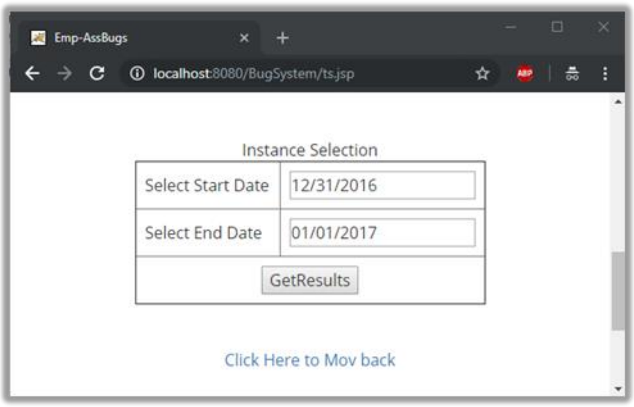


Fig. 4. Timestamp

*e. Checking Bug Intensity:*The bug intensity can be depicted in the form of a bar chart in 2 ways through our proposed system.

Algorithm 2: Intensity of bugs
Step 1: Select Employee analysis (EA) or performance analysis (PA).
Step 2: If EA then select an employee/developer E or goto step 5.
Step 3: Select the works done by E and score.
Step 4: Generate Final Report.
Step 5: If PA then select the total number of bugs raised in a projectand total number of bugs solved in a project.
Step 6: Generate Final Report.

- *Employee analysis*: In employee analysis, the total number of bugs that are solved by each developer is calculated. Firstly, the manager should select a developer from the list of developers as shown in Fig. 5(Developer Selection). Then the bar chart in Fig. 6 is produced based on the number of bugs that are solved by the developer selected in Fig. 5. The Y-Axis in Fig. 6 depicts the number of bugs that are solved by a particular developer.

From this, we can easily know how many bugs a particular developer solved so that we can assign the future bugs to the developer depending upon his/her bug solving intensity.

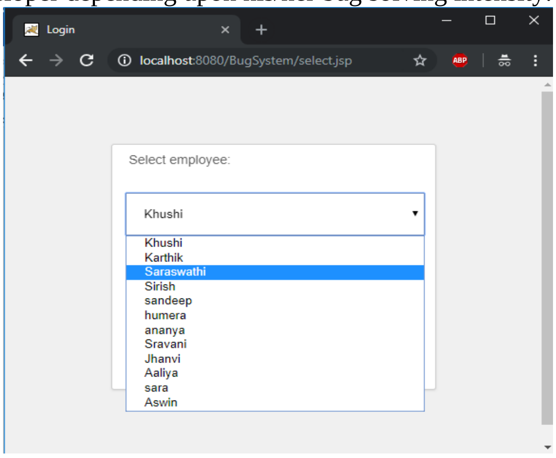


Fig.5. Developer Selection

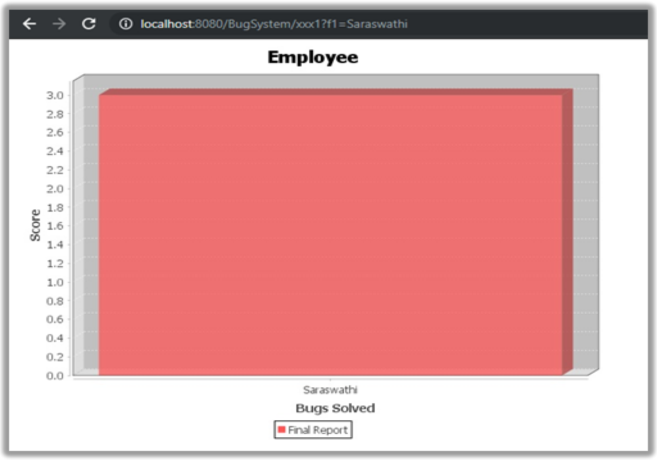


Fig.6. Employee Analysis

- *Performance analysis*: In performance analysis, the total number of bugs assigned to all developers and the total number of bugs solved by the developers are compared and a bar chart will be generated using the algorithm 3.

Algorithm 3: Performance Analysis
Raised=0;
Solution=0;
Resolved=0;
Bug = 0;

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

If (bug>0) //If any bug is raised
Raised++;
If (solution>0) //If any bug is solved
Resolved++;

For example for the data in Table. 4 shows that 5 bugs are resolved by the developers out of 7 bugs that are raised by the manager. Based on the values of the Table. 4 a bar chart will be generated.

Table. 4. Performance Analysis

Raised	Solved
7	5

B. EMPLOYEE MODULE

Employee tasks are: Updating assigned tasks, viewing bugs and raising questions in the blog.

a. Updating assigned tasks: Whenever bugs are assigned to a developer, they will be displayed one after the other to the developeronce he/she logins as shown in Fig. 7. Then the developer has to solve the bugs and update the bugs with their solutions.

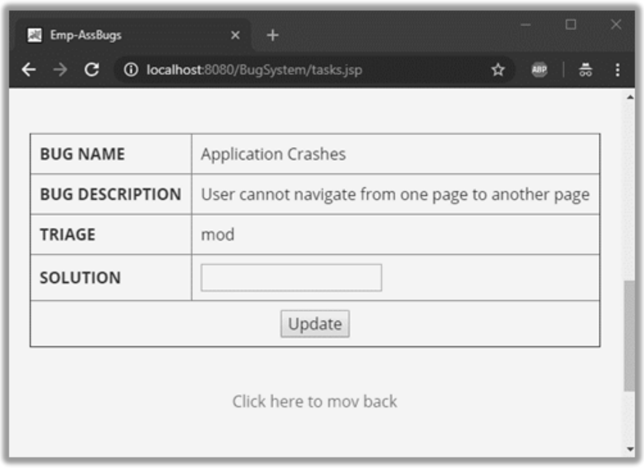


Fig.7. Assigned tasks

b. Viewing bugs: In our proposed system, the repository is open. So the bugs and the information about them are displayed to all the developers as shown in Fig. 8.

Fig. 8. Viewing Bugs

c. Raising Questions: A community forum is made for the developers to raise questions about the bugs which they can't solve as shown in Fig. 9.

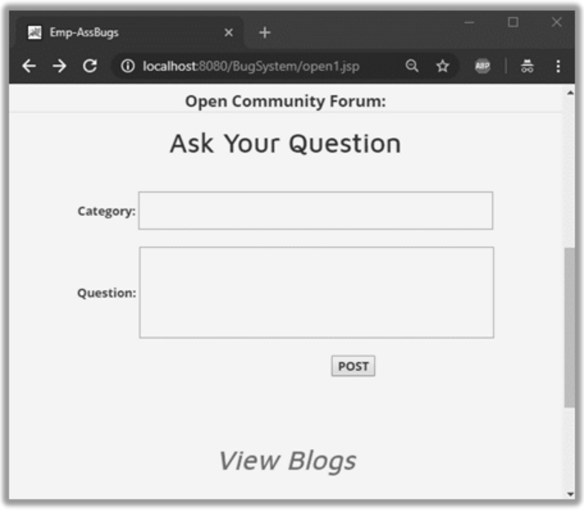
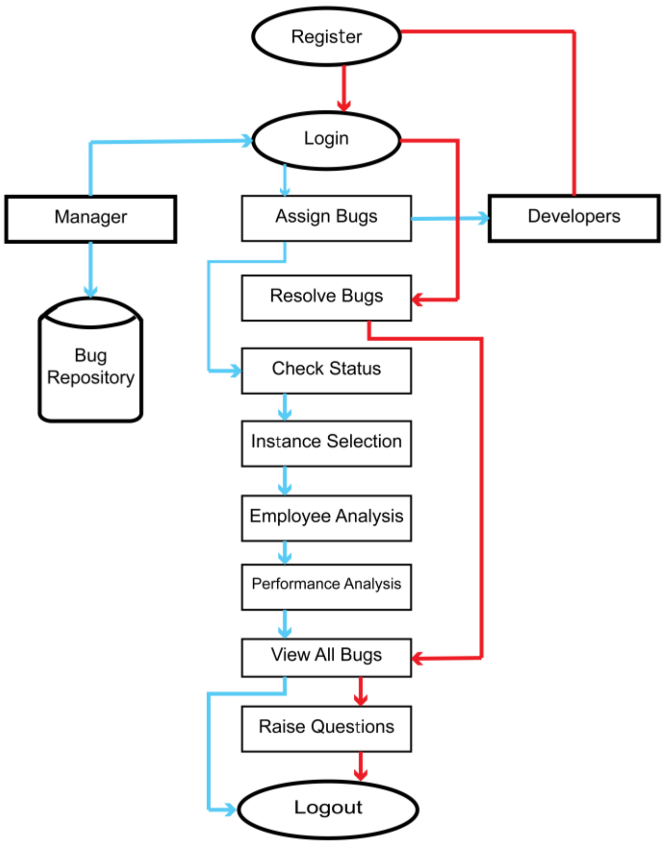


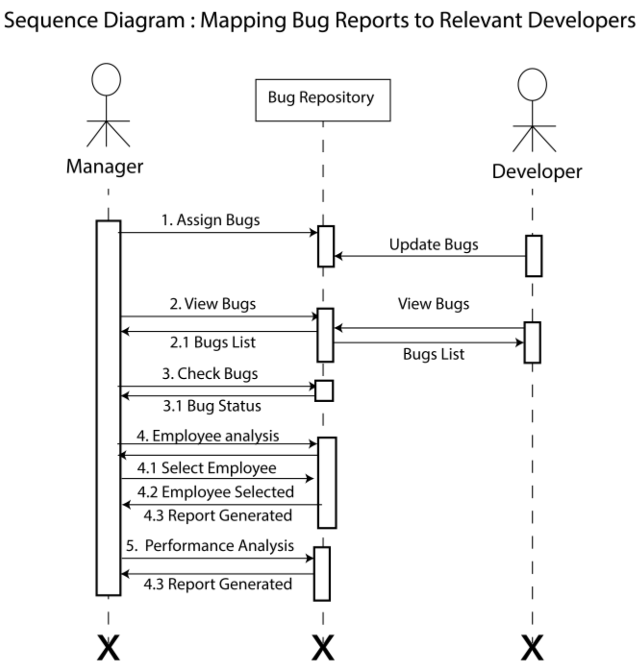
Fig.9. Raising Question

C. SYSTEM PROTOTYPE

Data Flow Diagram



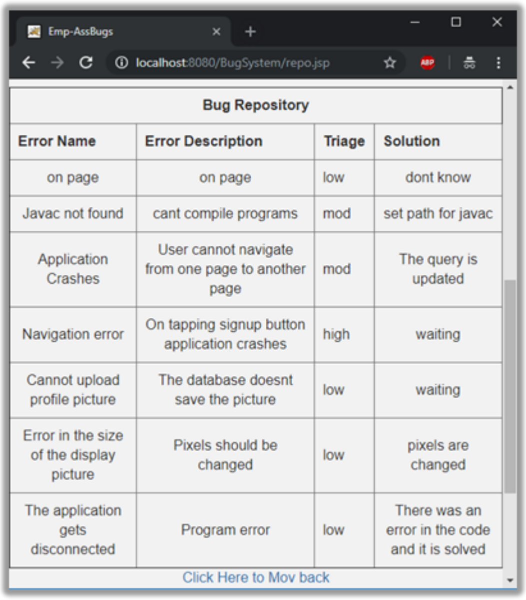
Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)



Bug Status				
Bug Name	Bug Description	Triage	Assigned To	Date
Application Crashes	User cannot navigate from one page to another page	mod	Saraswathi	01/01/2017
Navigation error	On tapping signup button application crashes	high	humera	01/01/2017
Cannot upload profile picture	The database doesnt save the picture	low	Sravani	01/01/2017
Error in the size of the display picture	Pixels should be changed	low	Saraswathi	01/01/2017
The application gets disconnected	Program error	low	Saraswathi	01/01/2017

Fig. 11. Instance selection

Since we have maintained a bug repository, we can easily find out the solution of the previously solved. which helps us to go through the bugs as shown in Fig.12.



Error Name	Error Description	Triage	Solution
on page	on page	low	dont know
Javac not found	cant compile programs	mod	set path for javac
Application Crashes	User cannot navigate from one page to another page	mod	The query is updated
Navigation error	On tapping signup button application crashes	high	waiting
Cannot upload profile picture	The database doesnt save the picture	low	waiting
Error in the size of the display picture	Pixels should be changed	low	pixels are changed
The application gets disconnected	Program error	low	There was an error in the code and it is solved

Fig. 12. Bug Repository

III. RESULTS AND DISCUSSION

The existing system [11] shows only those bugs that occurred recently in a particular project as shown below in Fig. 10.

Bug ID: 329950
Summary: "Close All" and "Close Others" may cause bundle activation.
Description: ...

Bug ID: 325722
Summary: "Close"-related context menu actions should show up for all stacks and apply to all items.
Description: ...

Bug ID: 313328
Summary: Close parts under stacks with middle mouse click.
Description: ...

Fig. 10. Existing System Bug History

But in our proposed system the bugs that occurred at any time instance can be generated as shown in Fig. 11. The time instance is selected as shown in Fig. 4 with the help of the Algorithm. 1: Instance Selection Algorithm the start and end dates are selected and the bugs obtained under that particular time. Since we preserved the details of bugs earlier, it is easy to fetch the repository and generate the bugs which occurred under a particular time period. From this, we can solve the bug easily if it occurred in the future as we preserve the bug report with a solution in bug repository.

With the help of the Algorithm. 3 and Table. 4 we can generate a bar chart as shown in Fig. 13 which shows the intensity of the bugs that how many bugs are solved. From Fig. 14 it's clear that the bugs with their solutions preserved in our implementation are more in number when compared to the existing system of mapping bug reports to

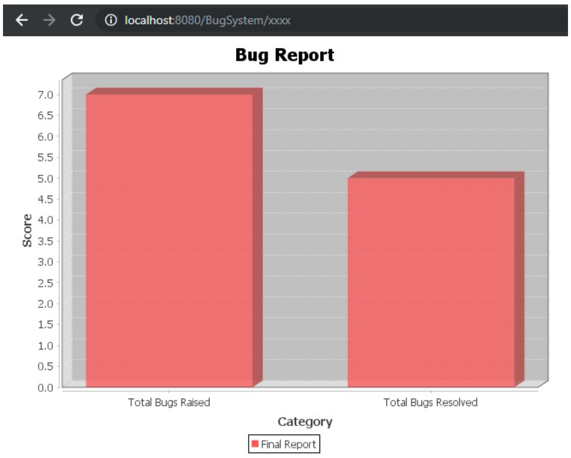


Fig. 13.Performance Analysis

relevant files. We save the bugs every time they are solved whereas in the existing system the bugs which occurred in the last month are saved and there will be no records saved about the bugs which occurred in a previous month. Preserving the information of bugs every time they are solved helps to review future bugs and solve them easily in the least time span.

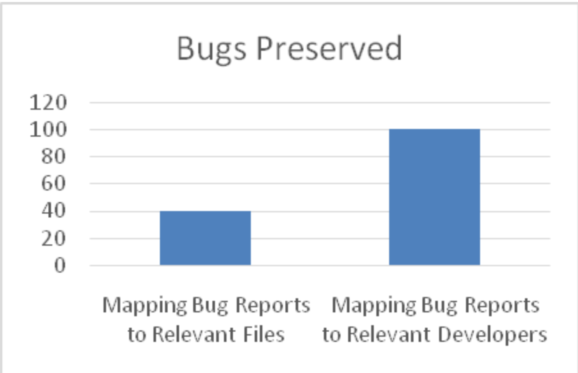


Fig. 14. Bar Chart

Advantages of the proposed system

- Mapping bug reports to relevant developers can advance the accuracy of the bugs.
- This proposed system automatically shows the intensity of the bugs.
- Maintaining a bug repository for all bug reports that are solved earlier deprecates time and expenses.
- The manpower required for this proposed system is very less.
- Time efficient and modern.

IV. CONCLUSION

The proposed system aims to reduce the time absorbing for solving bugs by assigning bugs to the developer based on their expertise. For example, a bug related to JAVA program will be assigned to a JAVA developer vice versa. So, the developers find it easy to find the bug and solve it. Preserving the status of the bug in bug repository helps us to easily solve the future bug by making an instance selection which saves a lot of time, manual work and cost. With the generation of the performance analysis bar chart, we can easily find out the intensity of the bug. Since the proposed system is implemented in JAVA, it is platform independent and can run on any system irrespective to the platform. Software companies working on huge projects can use the application of the proposed system.

Future Work

In future scope, we will leverage additional features like the grouping of developers and chat communication between developers.

REFERENCES

1. B. B and D. A.H., Object-Oriented Software Engineering Using UML, Patterns, and Java, NJ, USA: Prentice Hall, 2009.
2. R. B. P.L. and Z. Thomas, "Information Needs for software development analytics," in *34th International Conference on Software Engineering*, Zurich, Switzerland, 2012.
3. J. K and O. M, "Bug Prioritization to Facilitate Bug Report Triage," *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 1-5, 2012.
4. D. L. R. &. K. K. ShanthiPriya, "An Approach for Predicting Bug Triage using Data Reduction Methods," *International Journal of Computer Applications*, vol. 177, no. 5, pp. 1-5, 2017.
5. A. John, H. Lyndon and M. G. l C. , "Who Should Fix This Bug?," in *28th International Conference on Software Engineering*, Shanghai, China, 2006.
6. X. Jifeng , . J. He and . Z. Weiqin, "Developer prioritization in bug repositories," in *34th International Conference on Software Engineering (ICSE)*, NJ, USA, 2012.
7. A. S.K.S, "Bug Triaging: Profile Oriented Developer Recommendation," *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, vol. 2, no. 1, pp. 1-6, 2015.
8. X. Jifeng , . J. He, R. Zhilei , . Y. Jun and . L. Zhongxuan, "Automatic Bug Triage using Semi-Supervised Text Classification," in *22nd International Conference on Software Engineering and Knowledge Engineering* , NJ, 2010.
9. C. C and Z. Peixiang , "Towards graphical models for text processing," *Knowledge and Information Systems*, vol. 36, no. 1, pp. 1-21, 2013.
10. Z. Thomas, P. Rahul , . B. Nicolas, J. Sascha , . S. Adrian and W. Cathrin , "What Makes a Good Bug Report?," *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 618-643, 2010.
11. Y. Xin, B. Razvan and L. Chang, "Mapping Bug Reports to Relevant Files: A Ranking Model, a Fine-Grained Benchmark, and Feature Evaluation," vol. 42, no. 4, p. 383, 2016.
12. A. Shay , . K. Adam, . D. Julian, F. Tip, D. Danny, A. P and D. Michael, "Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol. 36, no. 4, pp. 1-21, 2010.
13. A. John and M. Gail, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," *ACM Transactions on Software Engineering and Methodology* , vol. 20, no. 3, 2011.
14. B. Krisztian, . A. Leif and d. R. Maarten , "Formal Models for Expert Finding in Enterprise Corpora," in *29th annual international ACM SIGIR conference on Research and development in information retrieval*, Washington, 2006.
15. W. Zou, Y. Hu, J. Xuan and H. Jiang, "Towards training set reduction for bug triage," in *IEEE 35th Annual Computer Software and Applications Conference*, NJ, 2011.
16. N. Fenton and S. Pfleeger, Software Metrics: A Rigorous and Practical Approach, Boston: PWS Publishing.
17. S. S and V. R, "IMPROVED APPROACH FOR PREDICTING THE BUG TRIAGE USING DATA REDUCTION METHODS," *International Research Journal of Engineering and Technology*, vol. 3, no. 3, pp. 1-6, 2016.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)