Holistic Evaluation Framework for Automated Bug Triage Systems: Integration of Developer Performance

Dr.V.Akila
Dept. of Computer Science and Engineering
Pondicherry Engineering College
Pondicherry, India

Dr.V.Govindasamy
Dept. of Information Technology
Pondicherry Engineering College
Pondicherry, India

***Abstract*—Bug Triage is an important aspect of Open Source Software Development. Automated Bug Triage system is essential to reduce the cost and effort incurred by manual Bug Triage. At present, the metrics that are available in the literature to evaluate the Automated Bug Triage System are only recommendation centric. These metrics address only the correctness and coverage of the Automated Bug Triage System. Thus, there is a need for user-centric evaluation of the Bug Triage System. The two types of metrics to evaluate the Automated Bug Triage System include Recommendation Metrics and User Metrics. There is a need to corroborate the results produced by the Recommendation Metrics with User Metrics. To this end, this paper furnishes a Holistic Evaluation Framework for Bug Triage System by integrating the developer performance into the evaluation framework. The Automated Bug Triage System is also to retrieve a set of developers for resolving a bug. Hence, this paper proposes Key Performance Indicators (KPI) for appraising a developer's effectiveness in contribution towards the resolution of the bug. By applying the KPIs on the retrieved set of developers, the Bug Triage System can be evaluated quantitatively.**

Keywords—Bug Management; Bug Triage; Recommendation Metrics; Key Performance Indicators; Developer Performance; Bug Resolution Time

I. INTRODUCTION

Open Source Software (OSS) is a commercial software where full access to the code for viewing, modification, and redistribution is granted to all the users by agreeing to a free-of-cost license. Bug management is a central component of the software maintenance of the OSS. Bug Management in OSS is usually performed using Bug Management Systems like Bugzilla. The new bugs that arise after the deployment of a new version of software are first reported to the Bug Management system. The new bugs are manually verified and important attributes like Component and Severity are fixed. Following this, the bugs are assigned to a developer for resolution by a human triager. In summary, Bug management comprises the following three activities: (i) Bug Triaging, (ii) bug assignment to the software developer for solution and (iii) solving of the bug. Software maintenance expenditure is about 50% of the overall expenditure of the software project. In OSS development, the expenditure translates to time. Bug Triaging comprises checking for validity of the bug, assigning priority, severity and assigning the bug to a correct software developer. Manual Bug Triaging is time- consuming and fault prone [1],[2],[3].

The bugs are reported to the Bug Management System. The reported bug is verified for validity and is assigned a new tag and is assigned to a developer. If the developer is unable to resolve the bug he may reassign the bug to a new developer. This activity is captured in a bug tossing graph. The summary that is in the bug report and the Bug Tossing graph serves as the basis for any Automated Bug Triage system. The metrics that are used to evaluate the Automated Bug Triage system are: (i) Accuracy (ii) Precision(iii) Recall and (iv) Mean Steps To Resolve. Precision is a better parameter for software developer recommendation because the cost of false recommendation is much higher than in search engine. Further, the Mean Steps to Resolve parameter encodes only the number of steps in the predicted path. While the reduction in the number of steps to resolve is required, it is also vital to compare how far the predicted path is similar to the original path. The structure and the ordering of nodes in the predicted path needs to be compared with that of the original path. Metrics based on graph edit distance were used for this purpose[4][5].

The evaluation of the Automated Bug Triage System is based only on the Recommendation metrics[6]. This paper integrates the developer performance in the evaluation framework of the Automated Bug Triage System. The quality of the developer extracted by the Automated Bug Triage System is evaluated by the Key Performance Indicators.

II. RELATED WORK

The related work has been studied with a perspective of metrics used for evaluation of the Automated Bug Triage System. The summary of the survey is depicted in the Table 1. It is observed from the survey that the Automated Bug Triage System is evaluated only with Recommendation Metrics. It is essential to integrate the User metrics into the evaluation process in order to build confidence in the Automated Bug Triage System. The user metrics are built over the Developer Performance. The following section shows the developer performance assessment incorporated in Open Source Systems.

TABLE I. SUMMARY OF THE SURVEY

Paper	Machine Learning	Information Retrieval				Bug Tossing Graph
	Accuracy	Precision	Recall	F-Measure	Mean Reciprocal Rank	Mean Steps to Resolve
2]	✓	-		-	-	✓
1]	✓	-		-	-	✓
3]	✓	-		-	-	✓
7]	✓	-		-	-	✓
8]	✓	-		-	-	-
9]	✓	-		-	-	-
10]		-		✓	✓	-
11]		✓		✓	-	-
12]		✓		-	-	-
13]		-		-	-	-
14]		-		-	-	✓

A. *Developer Performance Assessment*

Developer Performance Assessment is a necessity in identifying the strength and weakness of a developer, for career advancement, and fine tuning a business organization[15]. The contribution of a developer towards the software maintenance is quite different from a developer's contribution in developing a software product. Measuring a developer's contribution towards the maintenance of an OSS System is even more complicated. This complication is due to the fact that there are no explicitly assigned roles for the developers. However, there are different roles a developer may assume in the course of bug resolution.

The different roles that the developer may play in the bug resolution process are reporter of a bug, triager, commenter, and assignee [16]. In OSS, usually, there are metrics for evaluating the bug characteristics. These metrics focus on the program slicing characteristics of the bug like a number of lines of code affected by the bug and Cyclomatic Complexity of the bug. [17]. However, these metrics are underutilized in evaluating the Bug Triage System. Further, the developer's performance may be assessed based on Buggy commits, code contributions, and priority bugs. Buggy commits are used to identify developers who performed less buggy commits. Code contribution is measured regarding code addition, code removal, method addition, and method modification. The developer may also be assessed in terms of the number of high priority bugs that he has resolved [18]. In most of the existing works, developer's performance assessment is treated as an independent module. In the following section, the developer's performance assessment is integrated into the evaluation of the Bug Triage System. There are several KPIs proposed to assess the developer. These indicators are further utilized in quantifying the performance of the Bug Triage System.

B. *Key Observations from the Dataset*

This section gives a brief preview of the various factors that affect the bug resolution which is observed in the dataset. The bug reports of Eclipse project from www.bugzilla.org from 2009 to 2013 were analysed. The developers contribution for the various fields in the bug report like CC, status, Keywords, Summary priority, Assignee, and resolution are given in Figure 1. It is evident that 62% of the developers change the status of the bug to 'resolved'.

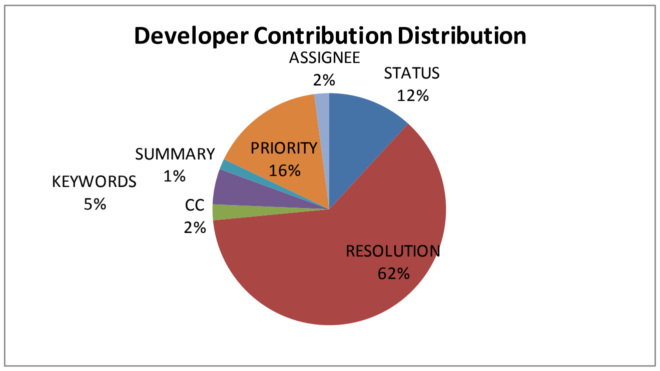

Fig. 1. Developer Contribution Distribution

The average time spent by a developer on a particular field of the bug report is given in Figure 2. As observed, the time spent to set the assignee field, status field and resolution field contributes mostly in the bug resolution time.

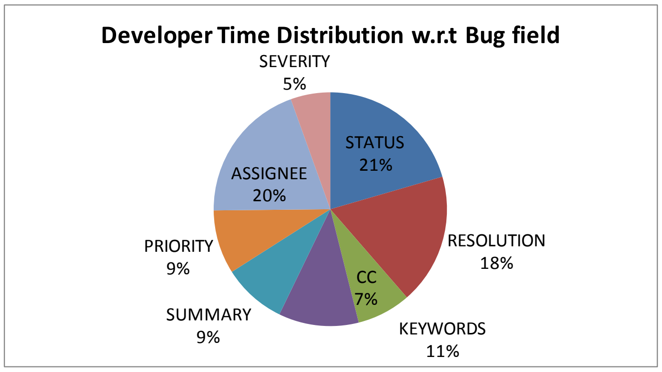

Fig. 2. Developer Time Distribution w.r.t Bug Field

The developer distribution with respect to the time spent by a developer on a particular bug is given in the Figure 3. It can be observed that 39% of the developers spend 121 to 700 days on a particular bug. Only 17% of the developers spend less than 6 months on a bug. Any Bug Triage System that extracts its set of developers mostly from this pool of 17% is a successful triage system.

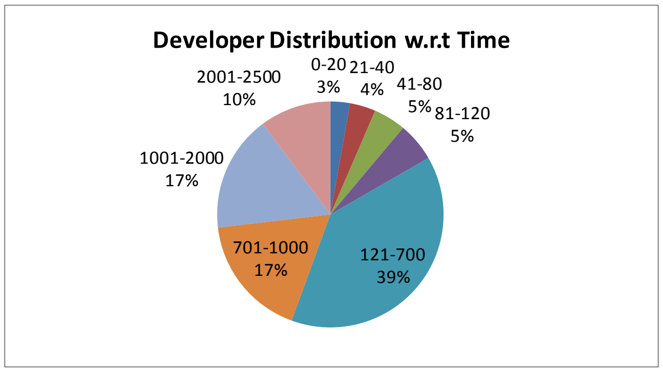

Fig. 3. Developer Distribution w.r.t Time

Figure 4 shows the Developer Distribution against the range of Bug Resolution Time. It can be observed that the most ineffective bug resolution is when the bug resolution time is more than 2 years. There are 44% of developers who spend time on bug whose resolution time is > 2 years. It can be observed from the chart that only 25% of developer has spent time in bugs that were resolved before six months. The motivation behind any Bug Triage System is to retrieve the developers from this pool of 25% of developers.

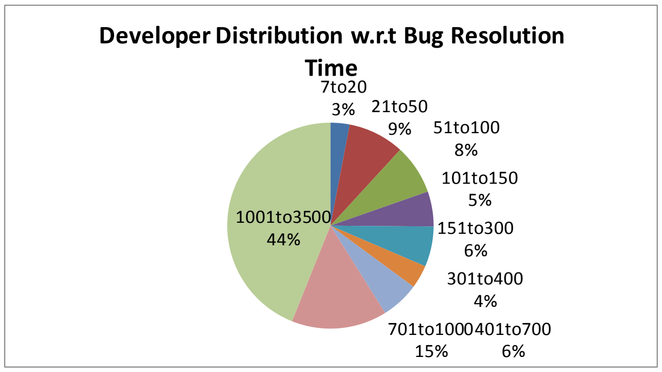

Fig. 4. Developer Distribution w.r.t Bug Resolution Time

Based on these observations, the Key Performance Indicators for assessing the Developer are introduced in the next section.

III. Key Performance Indicators for Assessing Developer Performance

The KPIs devised to evaluate the developer are Developer Time Index, Developer Effective index, and Developer Productivity. The Developer Time Index, Developer Effective index, and Developer Productivity are derived from Developer Contribution Count and Developer Contribution Time. The dependencies among the KPIs are depicted in Figure 5.

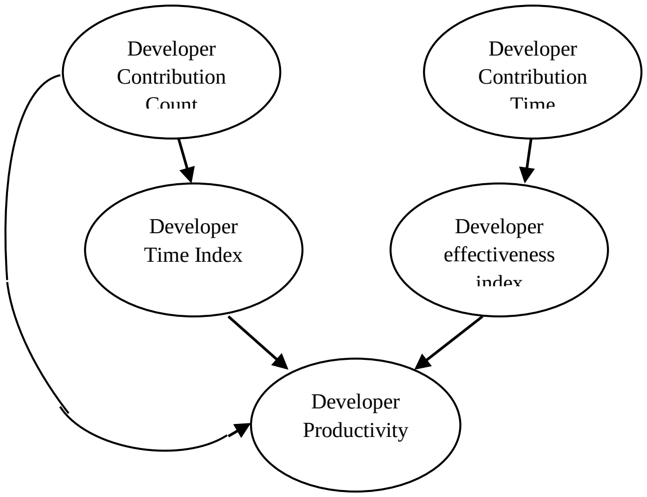
Developer Contribution Count
Developer Contribution Time
Developer Time Index
Developer effectiveness index
Developer Productivity

Fig. 5. Dependency in the Key Performance Indicators

The developer may have made the following types of contribution: CC, reassign the bug, change the Status field, or finally resolve the bug. For convenience sake, all the contributions are equally treated.

A. Developer Contribution Count

Developer Contribution Count (DCC) is defined as the number of contributions made by each developer in the process of resolving them.

$$\mathrm{DCC} = \sum_{i}^{n} C_i$$

where,

C_i - Contribution by a developer to a single bug.

n - Total number of bugs assigned to a developer

B. Developer Contribution Time

Developer Contribution Time (DCT) is defined as the time taken by each developer to make a contribution on a single bug.

$$\mathrm{DCT} = \forall\, \mathrm{Bug(Developer)} \sum(\mathrm{DBR} - \mathrm{DBA})$$

where, DBR- Date of Bug Reassignment

DBA – Date a Bug Assignment

C. Developer Time Index

Developer Time Index (DTI) is defined as the ratio of DCT to DCC. This indicator captures the amount of time taken by a developer to make a single contribution.

$$\mathrm{DTI} = \frac{\mathrm{DCT}}{\mathrm{DCC}}$$

D. *Developer Effectiveness Index*

The bug resolution time is considered to calculate the Developer Effectiveness Index (DEI). The intuition behind DEI is that, if a developer has contributed towards a bug that has been resolved with less time, then the developer's effectiveness is increased. Contrarily, if a developer has contributed towards a bug that has taken a long time to resolve, then the weight assigned to the developer is reduced.

TABLE II. WEIGHT ASSIGNMENT TABLE

Bug Resolution Time (in days)	Weights
7– 20	7
21-50	6
51-100	5
101-150	4
151-300	3
301-400	2
401-700	1
701-1000	-0.25
1001-3500	-0.50

The bugs for 10 years of Eclipse project were studied and the Resolution Time (RT) was extracted. RT varies from lower to higher values. RT was divided into nine ranges and their weights were assigned as given in Table 2. The highest weight is assigned to the range of Resolution Time that falls between 7 and 20 days. Negative weights are assigned to a range which took more than 700 days to resolve a bug.

The weights given here are inversely proportional to RT.

$$\text{Weight } (W_i) \alpha \frac{1}{RT}$$

DEI is defined as the ratio of the summation of Weights W_i of the bugs to the DCC.

$$\text{Developer Effectiveness Index DEI} = \frac{1}{DCC}\sum_{i}^{DCC} W_i$$

E. *Developer Productivity*

Developer Productivity (DP) is defined as the product of Developer Effectiveness Index, Developer Contribution Count, and the Developer Time Index.

$$DP = DEI * DTI * DCC$$

F. *A Holistic Evaluation Framework with Developer Performance*

The framework for evaluating the Bug Triage System is given in the Figure 6. The Bug Triage System extracts the optimal set of developers. KPIs of the retrieved developers are calculated and thereby, the Bug Triage System is assessed.

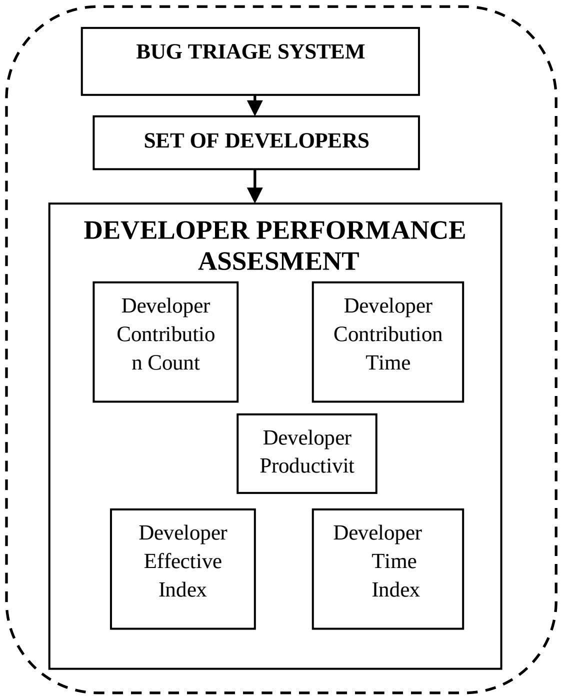
BUG TRIAGE SYSTEM
SET OF DEVELOPERS
DEVELOPER PERFORMANCE ASSESMENT
Developer Contribution Count
Developer Contribution Time
Developer Productivit
Developer Effective Index
Developer Time Index

Fig. 6. A Holistic Evaluation Framework with Developer Performance

IV. PERFORMANCE EVALUATION OF BUG TRIAGE SYSTEM WITH KPIS

The performance of the existing system GP-WBFS [2], BT-ANT [19] and the Multiple Ant Colony System (MACS) [20] were analysed using Developer Productivity, Developer Effectiveness and Developer Time Index. The Goal-oriented Path model with Weighted Breadth First Search (GP-WBFS) algorithm was compared only with Bug Triaging based on Ant System (BT-ANT) and the MACS because only in these systems adaptive learning was adopted. The graph for Developer Time Index is given in the Figure .7. It is evident from the Figure .7 that the Developer Time Index for the MACS as well as the BT-ANT is skewed towards Developer Time Index of<300. Almost 85% of the retrieved developers by MACS have a Developer Time Index of <300 and 65% of the developers retrieved by BT-ANT has a Developer Time Index of <300. Whereas in the existing GP-WBFS, 77% of the developers have a Developer Time Index >300.

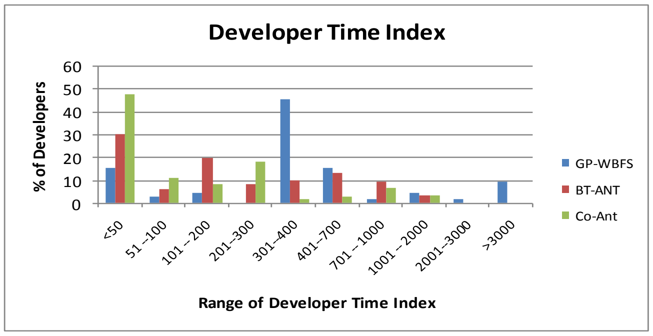

Fig. 7. Developer Time Index

The performance of the systems for Developer Effectiveness Index is given in Figure.8. Developer Effectiveness Index encodes the contribution of the developers for bugs that were resolved in a shorter period of time. From the graph, it is evident that 88% of the developers retrieved by MACS possess a Developer Effectiveness Index of >60 and 78% of the developers retrieved by the BT-ANT possess a Developer Effectiveness Index of >60. Whereas, in the developers retrieved by GP-WBFS, 78% of the developers have a Developer Effectiveness Index of <60.

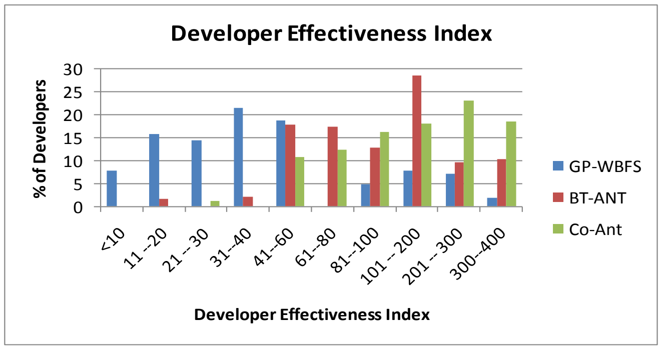

Fig. 8. Developer Effectiveness Index

The performance of the systems for Developer Productivity is given in Figure 9. Developer Productivity is a cumulative index that encodes the Developer Effectiveness, Developer Time Index and Developer Contribution Count. From Figure 9, it is evident that 91% of the developers retrieved by MACS possess a Developer Productivity of >50 and 75% of the developers retrieved by the BT-ANT possess a Developer Productivity of>50. Whereas in the developers retrieved by GP-WBFS, 73% of the developers have a Developer Productivity of <50.

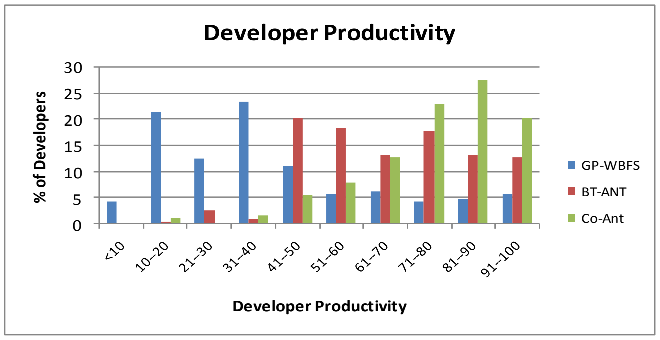

Fig. 9. Developer Productivity

V. Conclusion

This paper presents an Holistic Evaluation Framework for Bug Triage using developer performance. The metrics available in the literature that were used to evaluate the Bug Triage System were recommendation centric. The recommendation centric metrics evaluated the correctness and completeness of the recommendation mostly based on Precision and Recall measures. This paper adds a new dimension to the evaluation of the Bug Triage System. The evaluation framework factors in the quality of the developers extracted by the Bug Triage System in assessing the performance of the Bug Triage System. The evaluation metric based on the usefulness of the Bug Triage System is proposed. This is done by computing Key Performance Indicator values for the performance of the developers involved in the bug resolution. These calculated indices are then utilized to evaluate the developers extracted by the system. The proposed Key Performance Indicators are coarse grained in nature. A more fine grained analysis comprising the role analysis of the developers can be performed. The role analysis may be based on Social Network Analysis. Further , on the extracted roles of the developers more fine grained Key Performance Indicators are to be proposed.

Reference

[1] Pamela Bhattacharya and Iulian Neamtiu, "Fine-grained Incremental Learning and Multi-feature Tossing Graphs to Improve Bug Triaging," in IEEE International Conference on Software Maintenance, 2010, pp. 1-10.

[2] Pamela Bhattacharya, Iulian Neamtiu, and Christian R. Shelton, "Automated, Highly-accurate, Bug assignment using Machine Learning and Tossing Graphs," Journal of Systems and Software, vol. 85, no. 10, pp. 2275-2292, 2012.

[3] Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann, "Improving Bug Triage with Bug Tossing Graphs," in 7th Joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE '09), New York, NY, USA, 2009, pp. 111-120.

[4] V.Akila, G.Zayaraz "Bug Triage in Open Source System- A Review", International Journal of Collaborative Enterprise, Inderscience Publishers, vol.4, no.4, pp.299–319,2014

[5] V.Akila, G.Zayaraz, "Novel Metrics for Bug Triage", Journal of Software, vol. 9, no.12, pp.3035-3040, Dec 2014

[6] Iman Avazpour, Teerat Pitakrat, Lars Grunske, and John Grundy, "Dimensions and Metrics for Evaluating Recommendation Systems," in Recommendation Systems in Software Engineering.: Springer Berlin Heidelberg, 2014, pp. 245-273.

[7] Jifeng Xuan, He Jiang, Zhilei Ren, and Weiqin Zou, "Developer Prioritization in Bug Repositories," in 34th International Conference on Software Engineering (ICSE), Zurich, 2012, pp. 25 - 35.

[8] Wen Zhang, Song Wang, Ye Yang, and Qing Wang, "Heterogeneous Network Analysis of Developer Contribution in Bug Repositories," in International Conference on Cloud and Service Computing (CSC), Beijing, 2013 , pp. 98 - 105.

[9] Song Wang, Wen Zhang, Ye Yang, and Qing Wang, "DevNet: Exploring Developer Collaboration in Heterogeneous Networks of Bug Repositories," in ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, MD, 2013, pp. 193 - 202.

[10] Tao Zhang and Byungjeong Lee, "A Hybrid Bug Triage Algorithm for Developer Recommendation," in 28th Annual ACM Symposium on Applied Computing, Coimbra, Portugal, 2013, pp. 1088-1094.

[11] Geunseok Yang, Tao Zhang, and Byungjeong Lee, "Utilizing a Multi-Developer Network-based Developer Recommendation Algorithm to Fix Bugs Effectively," in 29th Annual ACM Symposium on Applied Computing (SAC '14)., Gyeongju, Republic of Korea, 2014, pp. 1134-1139.

[12] Tao Zhang and Byungjeong Lee, "An Automated Bug Triage Approach: A Concept Profile and Social Network Based Developer Recommendation," Intelligent Computing Technology,Lecture Notes in Computer Science, vol. 7389 , pp. 505-512, 2012.

[13] Shadi Banitaan and Mamdouh Alenezi, "DECOBA: Utilizing Developers Communities in Bug Assignment," in 12th International Conference on Machine Learning and Applications (ICMLA), Miami, 2013, pp. 66 - 71.

[14] Liguo Chen, Xiaobo Wang, and Chao Liu, "Improving Bug Assignment with Bug Tossing Graphs and Bug Similarities," in International

Conference on Biomedical Engineering and Computer Science (ICBECS), 2010 , Wuhan, 2010, pp. 1 - 5.

[15] Ayushi Rastogi, Arpit Gupta, and Ashish Sureka, "Samiksha: Mining Issue Tracking System for Contribution and Performance Assessment," in 6th India Software Engineering Conference, New Delhi,India, 2013, pp. 13-22.

[16] Tao Zhang, Geunseok Yang, Byungjeong Lee, and Ilhoon Shin, "Role Analysis-based Automatic Bug Triage Algorithm," Technical Report 2012.

[17] Raula Gaikovina Kula, Kyohei Fushida, Shinji Kawaguchi, and ajimu Iida, "Analysis of Bug Fixing Processes Using Program Slicing Metrics," in Product-Focused Software Process Improvement, Lecture Notes in Computer Science.: Springer Berlin Heidelberg, 2010, vol. 6156, pp. 32-46.

[18] Daniel Alencar da Costa, Uirá Kulesza, Eduardo Aranha, and Roberta Coelho, "Unveiling Developers Contributions Behind Code Commits: An Exploratory Study," in 29th Annual ACM Symposium on Applied Computing, Gyeongju, Republic of Korea, 2014, pp. 1152-1157.

[19] V.Akila, G.Zayaraz, V.Govindasamy, “Bug Triage based on Ant Systems”, International Journal of Bi- Inspired Computation , vol. 7 no. 4, pp. 263-268 , August 2015

[20] Govindasamy V, Akila V, Banu Priya, “Bug Triaging Using Multi-Attribute Bug Tossing Graph” Discovery Journal, 46(213), pp.101-106, 2015,