ISSN 2277-3061

UNDERSTANDING THE DEVELOPER PARTICIPATION IN BUG FIX PROCESS

[1]Madhu Kumari, [2]Meera Sharma and [3]Nikita Yadav*

[1]Delhi College of Arts & Commerce, University of Delhi,Delhi, India.
madhu_mesra@yahoo.com
[2]Swami Shraddhanand College, University of Delhi, Delhi, India.
meerakaushik@gmail.com
[3]Research Scholar, Singhania University, Pacheri Bari, Rajasthan.
aquanikita12@gmail.com

ABSTRACT

Prediction of the bug fix time in open source softwares is a challenging job. A software bug consists of many attributes that define the characteristics of the bug. Some of the attributes get filled at the time of reporting and some are at the time of bug fixing. In this paper, 836 bug reports of two products namely Thunderbird and Webtools of Mozilla open source project have been considered. In bug report, we see that there is no linear relationship among the bug attributes namely bug fix time, developers, cc count and severity. This paper has analyzed the interdependence among these attributes through graphical representation.

The results conclude that :

Case 1. 73% of bugs reported for Webtools are fixed by 17% developers and 61% of bugs are fixed by 14% developers for Thundebird.

Case 2. We tried to find a relationship between the time taken by a developer in fixing a bug and the corresponding developer. We also observed that there is a significant variation in bug fixing process, bugs may take 1 day to 4 years in fixing.

Case 3. There is no linear relationship between cc count i.e. manpower involved in bug fixing process and bug fix time.

Case 4. Maximum number of developers are involved in fixing bugs for major severity class.

Keywords

Open Source Software; Bug severity; Cc count; Bug fix time

Academic Discipline And Sub-Disciplines

Computer Science

SUBJECT CLASSIFICATION

Software Engineering

TYPE (METHOD/APPROACH)

Theory and Experimental Analysis

ISSN 2277-3061

INTRODUCTION

In open source software development process, software bug repositories provide crucial information to users/developers for the success of open source projects. An open source bug repository is the collection of bug reports that is available to the users/developers. The open source software provides source code of the software for further development and enhancement of the software. The data varies from versions to versions, change log data, web usage data, version archives, discussion forums on bug reports etc. The failure data is also maintained using different bug reporting and tracking system. The reported bugs contains many attributes such as severity, priority, components, operating system used, summary, description of the reports and status updates of the bug reports as time series. This data is very useful in conducting the research on software reliability, finding developer expertise, quality of software, resource utilization, effort, cost and time estimation, duplicate detection, dependency analysis, bug prediction, impact analysis, guiding co-change analysis, change prediction, and many more. To perform these analyses, we require the access to software repositories and analyze it which is called mining software repositories (MSR) [11].

A software bug has many attributes which are used to measure the quality and performance of the software. In this paper, we have considered the bug reports of two products of Mozilla open source software project. We have taken 4 quantified attributes namely bug fix time, cc count, developer id, and severity.

A software bug report is characterized by the following attributes[7].

Table 1 Bug Attributes description

Attribute	Short description
Severity	This indicates how severe the problem is. e.g. trivial, critical, etc.
Bug Id	The unique numeric id of a bug.
Priority	This field describes the importance and order in which a bug should be fixed compared to other bugs. P1 is considered the highest and P5 is the lowest.
Resolution	The resolution field indicates what happened to this bug. e.g. FIXED
Status	The Status field indicates the current state of a bug. e.g. NEW, RESOLVED
# Comments	Bugs have comments added to them by users. Number of comments made to a bug report.
Create Date	When the bug was reported.
Dependencies	If this bug cannot be fixed unless other bugs are fixed (depends on), or this bug stops other bugs being fixed (blocks), their numbers are recorded here.
Summary	A one-sentence summary of the problem.
Date of Close	When the bug was closed.
Keywords	The administrator can define keywords which you can use to tag and categorize bugs - e.g. The Mozilla Project has keywords like crash and regression.
Version	The version field defines the version of the software the bug was found in.
Cc Count	A list of people involved directly or indirectly in bug fix process. Who get mail when the bug changes.
Platform and OS	These indicate the computing environment where the bug was found.
# Attachments	Number of attachments for a bug.
Bug Fix Time	Last_Resolved time-Opened time. Time to fix a bug.

The rest of the paper is organized as follows. Section 2 of the paper describes the description of datasets. Results have been presented in section 3. Section 4 presents the related work and finally the paper is concluded in section 5.

DESCRIPTION OF DATASETS

We have taken bug reports of two products: Thunderbird (Client Software) for the period of april 2000 to march 2013 and Webtools (Server Software) for period of october 1998 to august 2013 of Mozilla open source software project. We Considered 221 bug reports of Thunderbird and 615 bug reports of Webtools . We collected bug reports for resolution “fixed” and status “verified”, “resolved” and “closed”. Some of the bug attributes are quantitative and some of them are qualitative in nature. So the qualitative bug attributes such as bug severity needs to quantify. We take from 1 to 7 for blocker to enhancement severity levels.

In this paper, we have taken 4 quantified bug attributes: bug severity, bug fix time, developer id, and cc count.

Bug severity. Bug severity is the degree of impact of the bug on the functionality of the software or product. In Mozilla open source software project seven different severity levels are defined.

Developer Id. A developer plays a major role in the software development process. If any bug is reported, it must be fixed by some developer to improve the development and performance of the software. Here, we assigned a number (1 to n) to each developer to do the analysis.

Cc Count. Manpower involved in monitoring the progress of bug fixing process.

Bug fix time. The time taken by a bug to get fixed, (Last resolved time – Opened Time).

RESULTS AND ANALYSIS

We considered three main cases to analyze the relationships between attributes of a bug.

In case 1 we show the distribution of bug count for developers who participated in fixing the bug , to analyse which developer have highest participation to fix the bug.

In case 2 we have taken first 25 developers and show the distribution of bug fix time for each developer.

In case 3 we show the distribution of Cc Count for each Bug fix time .

In case 4 . Maximum number of developers are involved in fixing bugs for major severity class.

Case 1 :

In this case we have taken 615 bug reports from Webtools product and we saw that 83 developers are involved in fixing of these bugs. But when we take those developer id's who fix more than 10 bugs , then the count of bug reports reduced 615 to 453 and the corresponding number of developers involved in fixing these bugs reduced from 83 to 14. This shows that 73 % of bugs are fixed by only 17% developers as shown in the figure 1.

Similar analysis we have done for Thunderbird product, we have taken 221 bug reports and saw that 49 developers are involved in fixing these bugs. But we take those developer id's who fix more than 10 bugs , then the count of bug reports reduced 221 to 134 and the corresponding number of developers involved in fixing these bugs reduced 49 to 7. This shows that 61% of bugs are fixed by only 14% developers as shown in the figure 2.

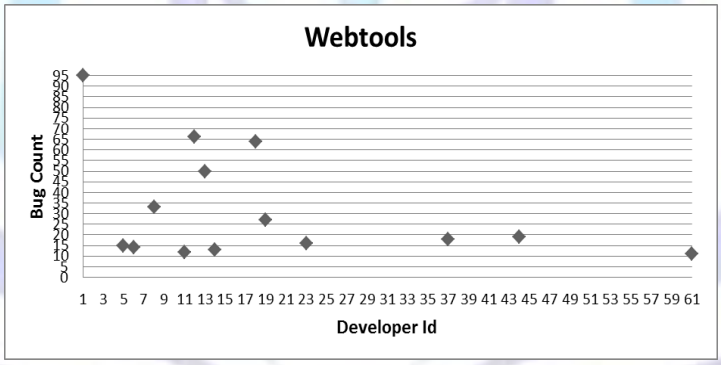


Figure 1. Distribution of bug counts by developer id for Webtools product.

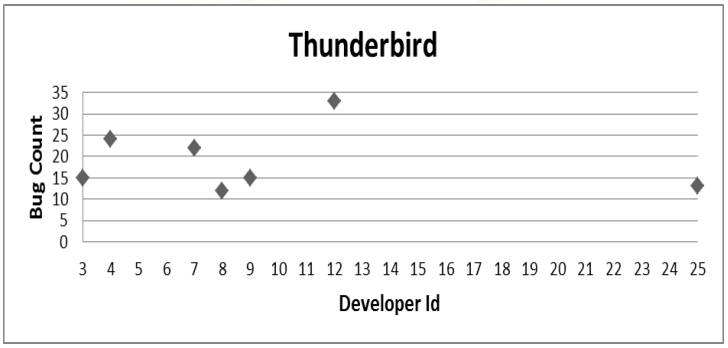


Figure 2. Distribution of bug counts by developer id for Thunderbird product.

Case 2:

We have analyzed the distribution of bug fix time for each developer and seen that it is very difficult to predict that how much time a bug will take to fix at the time of reporting. There is a significant variation in bug fixing process, bugs may take 1 day to 4 years in fixing.

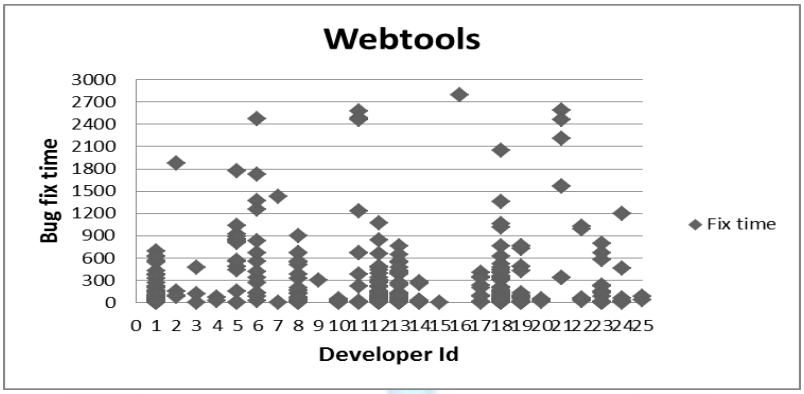


Figure 3. Distribution of bug fix time for each developer for Webtools product

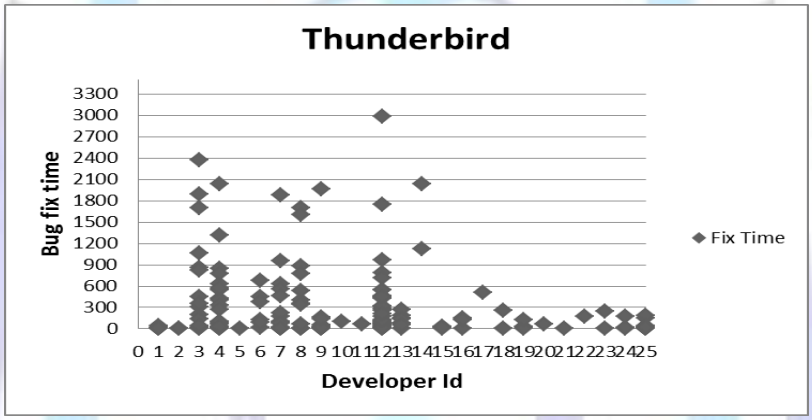


Figure 4. Distribution of bug fix time for each developer for Thunderbird product

Case 3:

We have analyzed the relationship between bug fix time and cc count for Webtools as shown in figure 3 and for Thunderbird as shown in figure 4.There is a large variation in bug fix time.

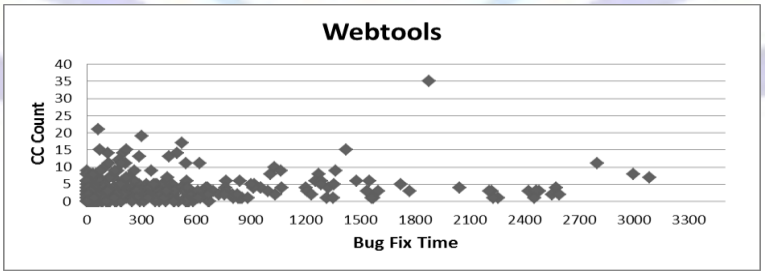


Figure 4. Analysis of Cc count and Bug fix time for Webtools product

ISSN 2277-3061

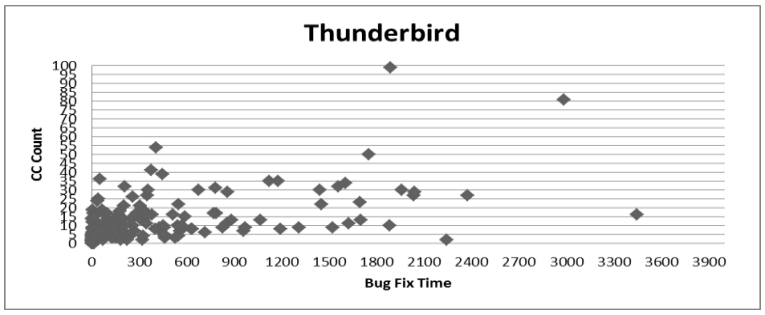
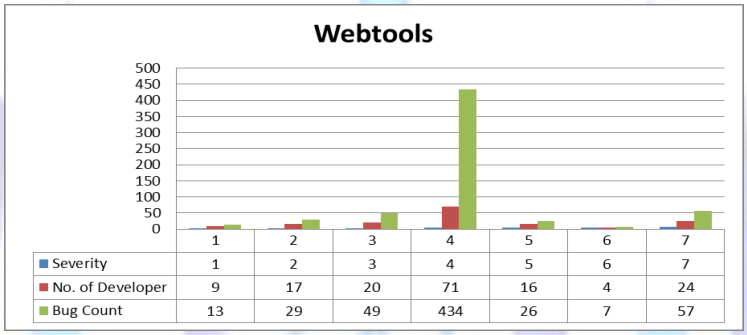


Figure 5. Analysis of cc list and Bug fix time for Thunderbird product.

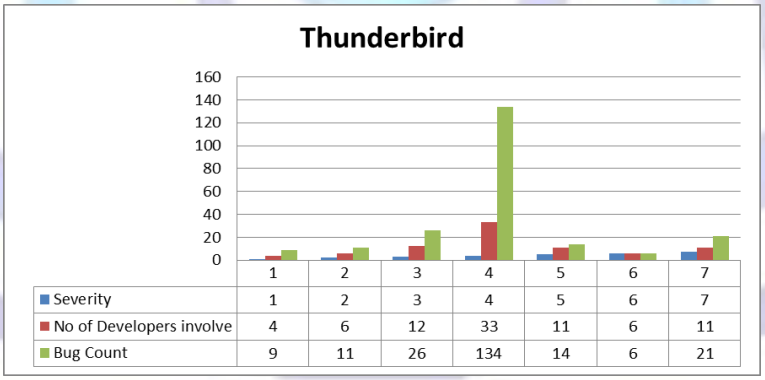
Case 4:

Here we computed how many developers are involved to fix in each bug severity and how many bugs are their in each severity .It has been shown in figure 4 and 5 the maximun developer involves to fix the most severe bug i.e major(4) and maximum number of bugs also lie in it.



	1	2	3	4	5	6	7
Severity	1	2	3	4	5	6	7
No. of Developer	9	17	20	71	16	4	24
Bug Count	13	29	49	434	26	7	57

Figure 6. Analysis of Bug severity corresponding to number of developers involve and bug count for Webtool product.



	1	2	3	4	5	6	7
Severity	1	2	3	4	5	6	7
No of Developers involve	4	6	12	33	11	6	11
Bug Count	9	11	26	134	14	6	21

Figure 7. Analysis of Bug severity corresponding to number of developers involve and bug count for Thunderbird product.

RELATED WORK

Bhattacharya and Neamtiu [9] proposed an idea of reducing tossing path lengths of a bug to 1.5-2 tosses for most bugs, which represents a reduction of up to 86.31% compared to original tossing paths. This reduction in tossing path length improved triaging accuracy and got 83.62% prediction accuracy in bug triaging.. They validated the approach on 856,259 bug reports of two software projects, Mozilla and Eclipse and 21 cumulative years of development . They have shown how intra-fold updates are beneficial for achieving higher prediction accuracy in bug triaging when using classifiers in isolation.

Bhattacharya and Neamtiu [3] used multivariate and univariate regression testing to test the prediction capability of existing models on 512,474 bug reports from five open source projects: Eclipse, Chrome and three products from the Mozilla project -Firefox, Seamonkey and Thunderbird. They have shown that the predictive power of existing models is between 30% and 49% so a room for more independent attributes is available. They demonstrate that, the bug-fix time in open source projects is not influenced by the bug-opener's reputation. They proposed that various bug report attributes which have been previously used to build bug-fix time prediction models do not always correlate with bug-fix time.

Sharma and Singh [7] used multiple linear regression analysis, support vector regression and fuzzy linear regression to show the contribution of bug attributes in predicting the cc list (the man power involved in monitoring the progress of bug fix)for a reported bug .They conducted the experiments on for 21,424 bug reports of Firefox, Thunderbird, Seamonkey, Boot2Gecko, Add-on SDK, Bugzilla, Webtools and addons.mozilla.org products of the Mozilla open source project.

Currently, Tian et al. [10] proposed a new approach to predict severity of a bug automatically in particular BM25-based document similarity function. They focused on predicting fine-grained severity labels, namely the different severity labels of Bugzilla . They proposed a new approach ,automatically analyzes bug reports reported in the past along with their assigned severity labels, and recommends severity labels to newly reported bug reports .

Kim and Whitehead [8] demonstrates the distribution of bug counts for each bug fix time .They computed and analyzed the bug fix time of files in ArgoUML and PostgreSQL by identifying when bugs are introduced and when the bugs are fixed.

CONCLUSION

Advancement in internet and communication technologies has eased the work process in distributed environment.The development of open source has got an edge due to the advancement in these technologies. The quality of software depends upon how much it satisfies the users requirements and at the same time without any failure. The present study , which focuses on reported bugs and their attributes of two software components will help in understanding the development of open source software. This study will help in improving the the software quality by understanding the relation among different bug attributes.The main findings are as follows:

Case 1. 73% of bugs reported for Webtools product are fixed by 17% developers and 61% of bugs are fixed by 14% developers for Thundebird product.

Case 2. We tried to find a relationship between the time taken by a developer in fixing a bug and the corresponding developer. We also observed that there is a significant variation in bug fixing process, bugs may take 1 day to 4 years in fixing.

Case 3. There is no linear relationship between cc count i.e. manpower involved in bug fixing process and bug fix time.

Case 4. Maximum number of developers are involved in fixing bugs for major severity class.

REFERENCES

[1] J. Bevan, E. J. Whitehead, Jr., S. Kim, and M. Godfrey,"Facilitating Software Evolution with Kenyon," Proc. of the 2005 European Software Engineering Conference and 2005 Foundations of Software Engineering (ESEC/FSE 2005), Lisbon, Portugal, pp. 177-186, 2005.

[2] D.Cubranic and G. C. Murphy , "Hipikat: Recommending pertinent software development artifacts," Proc. of 25th International Conference on SoftwareEngineering (ICSE), Portland, Oregon, pp. 408-418, 2003.

[3] Bhattacharya, P. and Neamtiu, I. 2010. Bug-fix Time Prediction Models: Can We Do Better? In Proceedings of the 8th Working Conference on Mining Software Repositories (New York, NY, USA,2012). ACM, 207-210. DOI= http://dl.acm.org/10.1145/1985441.1985472.

[4] M. Fischer, M. Pinzger, and H. Gall , "Populating a Release History Database from Version Control and Bug Tracking Systems," Proc. of 2003 Int'l Conference on Software Maintenance (ICSM'03), pp. 23-32, 2003. (Indore, India,2012). IEEE, 378-387. DOI= http://ieeexplore.ieee.org/ 10.1109/CONSEG.2012.6349519.

[5] A. Mockus and L. G. Votta, "Identifying Reasons for Software Changes Using Historic Databases, "Proc. of International Conference on Software Maintenance (ICSM 2000), San Jose, California, USA, pp. 120-130, 2000.

[6] J. Sliwerski, T. Zimmermann, and A. Zeller, "When DoChanges Induce Fixes?" Proc. of Int'l Workshop on Mining Software Repositories (MSR 2005), Saint Louis, Missouri, USA, pp. 24- 28, 2005.

[7] Sharma Meera, Kumari Madhu, and Singh VB, “Understanding the Meaning of Bug Attributes and Prediction Models” I-CARE '13 Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop Article No. 15 ,ACM New York, NY, USA ©2013

S. Kim and E. J. Whitehead, Jr. How long did it take to fix bugs? In MSR, 2006.

[9] Bhattacharya, P. and Neamtiu, I. 2010. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. In *Proceedings of the International Conference on Software Management*(Washington, DC, USA, 2010).ACM, 1-10. DOI= http://dl.acm.org/10.1109/ICSM.2010.5609736.

[10] Tian, Y., David L., and Sun, C. 2012. Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction. In *Proceedings of the 19th Working Conference on Reverse Engineering (WCRE),*(15-18 Oct. 2012). 215-224.

[11] Chaturvedi, KK, Singh VB and Singh Prashasht, Tools in Mining Software Reposotories,2013 13th International Conference on Computational Science and Its Applications, IEEE Explore, pp.89-98