Prediction model for bug re-opens in Mozilla Firefox

Prabhdeep Kaur*
Department of Computer Engineering,
Baba Banda Singh Bahadur Polytechnic College
Fatehgarh Sahib,Punjab,India

Puneet Mittal[2]
Department of Computer Science and Engineering,
Baba Banda Singh Bahadur Engineering College
Fatehgarh Sahib,Punjab,India

Amandeep Singh[3]
Department of Computer Engineering,
Baba Banda Singh Bahadur Polytechnic College
Fatehgarh Sahib,Punjab,India

***Abstract:* - Bug fixing becomes the most crucial activity in the software development process. Fixing bugs accounts a large amount of time of software development task. Sometimes these bug fixes are incomplete or inappropriate and results in bug reopen. Reopened bugs degrades the overall quality of the software perceived by the users and also increases the maintenance costs and indicates instability in the software system. Bugs can be re-opened for a variety of reasons. In this paper, we determined that which factors indicate whether a bug will be re-opened or not using Bugzilla database for Mozilla Firefox. An analysis is performed on components, different severity levels and the last resolution for the trunk version and Unspecified version of Mozilla Firefox to study their impact on bug re-opens. A relationship between significant factors is also established to make the prediction model more accurate. The findings of this work contribute towards better understanding of what factors impact bug re-openings so they can be examined more carefully.**

Keywords: - Bug re-open, Component, Severity, Last Resolution.

I INTRODUCTION

Every developer wants to develop software which is free from errors or bugs. But as a man-made artifact, software suffers from various software bugs, which cause crashes, hangs or incorrect results and significantly threaten the reliability and also the security of computer systems. Software Quality cannot be improved without knowledge of development process. The number of bugs and errors occurred during the software development process have to be found in the early stages of development for better quality. Bugs are detected either during testing before release or in the field by customers post-release. Once a bug is discovered, developers usually need to fix it. In particular, for bugs that have direct, severe impact on customers, vendors usually make releasing timely patches the highest priority in order to minimize the amount of system down time [29].

Unfortunately, fixes to bugs are not bullet proof since they are also written by human. Some fixes either do not fix the problem completely or even introduce new problems. Mistakes in bug fixes may be caused by many possible reasons. First, bug fixing is usually under very tight time schedule, typically with deadlines in days or even hours, definitely not weeks. Such time pressure can cause fixers to have much less time to think cautiously, especially about the potential side-effects and the interaction with the rest of the system. Similarly, such time pressure prevents testers from conducting thorough regression tests before releasing the fix.

Second, bug fixing usually has a narrow focus (e.g., removing the bug) comparing to general development. As such, the fixer regards fixing the target bug as the sole objective and accomplishment to be evaluated by his/her manager. Therefore, he/she would pay much more attention to the bug itself than the correctness of the rest of the system [29]. Bug reopening is of vital interest to the software developers in order to

- Improve the quality of bug fixing process
- Identify important issues that are not fixed and later result in bug reopens
- Identifying factors that influence the likelihood of a bug being re-opened
- Minimize the occurrence of re-opened bugs.

II RELATED WORK

The work closest to this study is by Zimmermann et al. [1] who characterized the bug reopen process using a mixed methods approach: they qualitatively identified causes for bug reopens based on the survey responses of Microsoft engineers and performed a quantitative analysis using bug reports from the Windows operating system to assess the impact of the various factors. The findings focusing on factors related to bug report edits and relationships between people involved in handling the bug. Finally, they build statistical models to describe the impact of various metrics on reopening bugs ranging from the reputation of the opener to how the bug was found.

More related work is done by Shihab et al. [2] who predicted reopened bugs in the Eclipse project. They used measures from four dimensions—work habits, bug report, bug fix and team—as input for decision trees (C4.5), which predicted reopened bugs with a precision of 62.9% and a recall of 84.5%. With a *top node analysis* they found that the bug report dimension was most influential. In addition to the work by Shihab et al. [2], this study includes a strong qualitative component on the *causes* of bug reopens (identified through survey comments) and also presents complete descriptive analysis. While Shihab et al. [2] used decision trees which are descriptive too, they only presented the trees aggregated to the top nodes in their paper.

Several other studies modeled the lifetimes of bugs, investigating properties like time-to-resolve (how long it takes a bug report to be marked as *resolved*), where the resolution can be of any outcome (e.g., FIXED, WON'TFIX, DUPLICATE, WORKSFORME). Hooimeijer and Weimer [3] built a descriptive model for the lifetime of a bug report based on selfreported severity, readability, daily load, reputation, and changes over time. This model shows that it could reduce software maintenance costs if the average cost of triaging a bug report is greater than 2% of the cost of ignoring an important issue. Panjer [4] explored the Eclipse bug set with various data mining algorithms reveal that an accuracy of 34.9% can be achieved using only the primitive attributes associated with a bug. They used information known at the beginning of a bug's lifetime such as severity, component, platform, and comments to predict its time-to-resolve. Bettenburg et al. [5] observed that bug reports are fixed sooner when they contain stack traces or are easy to read.

Anbalagan and Vouk [6] found that the more people are involved a bug, the higher its time-to-resolve. Mockus et al. [7] found that in Apache and Mozilla, bugs with higher priority are fixed faster than bugs with lower priority. Herbsleb and Mockus [8] observed that distributed work items (e.g., bug reports) take about 2.5 times as long to resolve as co-located work items. Cataldo et al. [9] found that when coordination patterns are congruent with their coordination needs, the resolution time of modification requests (similar to bug reports) was significantly reduced. In contrast to these time-to resolve studies, this study analyze when bug reports are reopened.

Several studies characterized properties of bug reports and their edit activities: Bettenburg et al. [5] characterized what makes a good bug report. Aranda and Venolia [10] examined communication between developers about bug reports at Microsoft to identify common bug fixing coordination patterns. Breu et al. [11] categorized questions asked in open-source bug reports and analyzed response rates and times by category. Bettenburg et al. [12] quantified the amount of additional information in bug duplicates. Jeong et al. [13] analyzed the reassignment of bug reports (called bug tossing) and developed tossing graphs to support bug triaging activities.

Kim et al. [14] computed the bug-fix time of files in ArgoUML and PostgreSQL by identifying when bugs are introduced and when they are fixed. They reported two bug-fix time statistics: average bug-fix time, and files whose bug-fix time were above average and suggested that the files which took above average time to fix should be refactored. Giger et al. [15] studied six projects: Eclipse JDT, Eclipse Platform, Mozilla Core, Mozilla Firefox, Gnome GStreamer and Gnome Evolution. They found that using post-submission data of bug reports (i.e., number of comments made to a bug and number of developers involved) improves bug-fix time prediction accuracy. Additionally their model could predict how promptly a new bug report will receive attention. They measured how attributes used by these prediction models correlate with bug-fix time, and found correlation values to be low. Ko et al. [16] conducted a linguistic analysis of bug report titles and observed a large degree of regularity. Bertram et al. [17] conducted a qualitative study of issue tracking systems as used by small, collocated software development teams. They found that even in collocated teams, issue trackers are a focal point for communication and coordination. Ko and Chilana [18] quantified the value of contributions by "power users" to open bug reporting in Mozilla. They observed that the primary value comes from recruiting a small pool of talented developers and reporters, and not from the masses.

Anvik et al. [19] presented an approach to semi- automating the assignment of a bug report to a developer with the appropriate expertise to resolve the report. Their approach uses a supervised machine learning algorithm that is applied to information in the bug repository. In addition to presenting their approach and results, they have presented an in-depth analysis of the application of machine learning to the problem and they have reported on lessons learned in trying to make use of data in the bug repository. To improve bug triaging, previous research

proposed techniques to semi-automatically assign developers to bug reports [20, 21], assign locations to bug reports [22], recognize bug duplicates [23,24,25,26,27], assess the severity of bugs .

This work adds a characterization of what bug reports are reopened to that body of knowledge. In this study, it is discovered that which factors can accurately predict the probability of a bug to be reopened and to find out the relationship between those factors.

III METHODOLOGY

To predict the reopeness of a bug and to identify the factors that influence the likelihood of a bug being re-opened, Mozilla Firefox [31] was studied because it is large and mature open source software.

To conduct this study, Bugzilla [30] database for Mozilla Firefox was evaluated. Then we extracted the reopened bug reports from Bugzilla database for Mozilla Firefox. The Bugzilla database contains all bugs that have been found in the lifetime of the Mozilla Firefox project with the detailed information that includes the release number, bug severity, and summary of bugs. Bugzilla stores the bugs in SQL database, so extracting the bugs from it was a straightforward task. Then the manual examination of the reopened bug report was performed. The following fields were extracted from the bug reports:

- Bug ID: Unique Identity of the bug.
- Bug Reporter: Who reported this bug?
- Component: The component in which bug is found.
- Assignee: Who is responsible to handle the bug?
- Last Status: The Status when the bug was closed before it reopened.
- Times the bug reopened: The number of times a bug was reopened in its life time.
- Last Resolution: How has the bug been resolved?
- Version: Version in which bug was found.
- Severity: Severity of the reopened bug (High, Medium and Low).
- Platform: The hardware specification in which bug has been found.

After collecting and analyzing the reopened bug reports, it was found that the bugs of Trunk and Unspecified versions had maximum reopens. So, these two versions of Mozilla Firefox were considered in this study.

When the required fields were extracted from the bug reports, analysis of data was performed to characterize which factors influence the bug reopen rate. Only those fields were considered in this study which had impact on bug reopens, other fields were discarded.

Then the data was analyzed to find out their impact on bug re-opens by calculating the reopen rate of different factors. Only those factors were considered in study which has high reopen rate and more chances to re-open and those were discarded which has no significant effect on bug re-opens. After that accuracy of prediction model was evaluated by calculating re-open precision.

IV RESULTS

In this section, the results of analysis performed on collected data to discover which factors influence the reopen probability of a bug are presented. To do this, statistical analysis of data was performed. After collecting and analyzing the reopened bug reports, it was found that the bugs of Trunk version and Unspecified version has higher probability to be reopened.

Table 1: Bug report data statistics

	Trunk	Unspecified
Total extracted bug reports	5747	5907
Reopened bug reports	475	395
Not reopened bug reports	5272	5512

Table 1 shows the number of bug reports used for each version. We had extracted 5747 bug reports for Trunk Version. Of these 5747 reports only 475 bug reports were reopened and 5272 were not. For each bug report the required fields were extracted to perform the analysis to describe which factors has impact on bug reopens.

Table 2 shows the components statistics for the Trunk and Unspecified version. It is observed that some components have high reopen rate as compared to other i.e. the component in which the bug was found has much impact on bug reopens. The following components have high reopen rates:

- Developers Tool
- General
- Session Restore
- Disability Access
- Private Browsing
- Build Config
- Panorama
- PDF Viewer

Table 2: Statistical analysis based upon component

Component	Reopen Rate Trunk	Reopen Rate Unspecified
Developers Tool	0.174	0.092
General	0.101	0.089
Session Restore	0.092	0.078
Disibility Access	0.091	0.085
Private Browsing	0.078	0.077
Build Config	0.074	0.068
Panorama	0.071	0.068
PDF Viewer	0.068	0.051
Theme	0.060	0.043
Tabbed Browser	0.054	0.063
File Handling	0.053	0.042
Keyboard Navigation	0.052	0.075
Toolbars and Customization	0.048	0.039
Menus	0.047	0.039
Downloads Panel	0.043	0.023
Location Bar	0.041	0.051
Search	0.037	0.065
Security	0.037	0.006
Bookmarks and History	0.033	0.012
RSS Discovery and Preview	0.019	0.024
Preferences	0.019	0.025
Shell Integration	0.010	0.007

When the severity data was analyzed, it was discovered that severity has a significant impact on the bug reopens as shown in the Table 3. The high severity bugs have more probability of reopening.

Table 3: Statistical analysis based upon severity

Severity	Reopen Rate Trunk	Reopen Rate Unspecified
High	0.260	0.247
Medium	0.073	0.061
Low	0.091	0.051

Table 4 indicates the probability of a bug being reopened based upon the last resolution of the reopened bug. The results show that last resolution has influence on bug being reopened. Hence it is an important factor for our

study. The reopen bug with the last resolution DUPLICATE, FIXED and WORKSFORME has high reopen rates.

Table 4: Statistical analysis based upon last resolution

Last Resolution	Reopen Rate Trunk	Reopen Rate Unspecified
DUPLICATE	0.084	0.057
FIXED	0.086	0.078
INCOMPLETE	0.044	0.080
INVALID	0.052	0.047
WONTFIX	0.078	0.054
WORKSFORME	0.089	0.082

In this study, a relationship between factors was also drawn which can accurately predict bug reopen as indicated in Table 5.

Table 5: Relationship between component and severity for Trunk version

Component	High Severity Reopen Rate	Medium Severity Reopen Rate	Low Severity Reopen Rate
Developers Tool	0.667	0.130	0.926
General	0.541	0.076	0.068
Private Browsing	0.500	0.042	0.083
Panorama	0.200	0.078	0.042
Session Restore	0.167	0.084	0.094
Tabbed Browser	0.143	0.059	0.034
Toolbars and Customization	0.143	0.061	0.000
Bookmarks and History	0.063	0.034	0.023
Disability Access	0.000	0.059	0.200
Build Config	0.000	0.070	0.125
PDF Viewer	0.000	0.067	0.091
Theme	0.000	0.052	0.103
File Handling	0.000	0.068	0.037
Keyboard Navigation	0.000	0.065	0.036
Menus	0.000	0.050	0.043
Downloads Panel	0.000	0.048	0.000
Location Bar	0.000	0.045	0.038
Search	0.000	0.039	0.031
Security	0.000	0.033	0.056
Migration	0.000	0.053	0.000
Untriaged	0.000	0.034	0.000
RSS Discovery and Preview	0.000	0.024	0.000
Preferences	0.000	0.017	0.024
Shell Integration	0.000	0.013	0.000

Table 5 shows the relationship of the component in which the bug was found with the severity of bug being reopened. It is clear from the table that High severity bugs in the components Developers Tool, General, Private Browsing, Panorama, Session Restore, Tabbed Browser, Toolbars and Customization has more probability to reopen. Similarly in the components Downloads Panel, Location Bar, Search, Migration, Untriaged, RSS Discovery and Preview and Shell Integration, Medium severity bugs have more reopen rate. Whereas in Disability access, Build config, PDF viewer, Theme, Security and Preferences, the reopen rate of Low severity bugs is high.

When this analysis was applied on Unspecified version, the results that appeared are shown in Table 6

Table 6: Relationship between component and severity for Unspecified version

Component	High Severity Reopen Rate	Medium Severity Reopen Rate	Low Severity Reopen Rate
Disability Access	1.000	0.053	0.125
Private Browsing	1.000	0.040	0.077
Developers Tool	0.750	0.077	0.373
Location Bar	0.500	0.057	0.021
General	0.495	0.065	0.031
Menus	0.200	0.029	0.034
Keyboard Navigation	0.143	0.090	0.043
Toolbars and Customization	0.125	0.042	0.020
Session Restore	0.063	0.078	0.091
Build Config	0.000	0.083	0.000
Panorama	0.000	0.073	0.050
Search	0.000	0.074	0.040
Tabbed Browser	0.000	0.071	0.045
Social API	0.000	0.059	0.000
PDF Viewer	0.000	0.040	0.125
Theme	0.000	0.038	0.077
File Handling	0.000	0.063	0.023
Page Info Window	0.000	0.059	0.000
Preferences	0.000	0.035	0.011
RSS Discovery and Preview	0.000	0.029	0.000
Download Panel	0.000	0.028	0.000
Sync	0.000	0.020	0.000
Extension Compatibility	0.000	0.018	0.000
Bookmarks and History	0.000	0.014	0.011
Shell Integration	0.000	0.012	0.000
Security	0.000	0.012	0.000

After analyzing the data it was discovered that in the components Disability Access, Private browsing, Developers Tools, Location Bar, General, Menus, Keyboard Navigation and Toolbars and Customization, the bugs with High severity had more chances to be reopened. Whereas Medium severity bugs had high reopen rate in Build config, File handling, Social API, Tabbed browser, Preferences, Download panel, Sync, security and Page info window. Whereas in Theme, Session restore and PDF viewer, Low severity bugs had high reopen rate.

A relationship is also discovered between component in which a bug was and the last resolution of the bug when it was closed as shown in table 7. Following components had high reopen rate in the last resolution field shown against them:

DUPLICATE- Private Browsing, Download Panel, Developers Tool, File Handling, Build Config, Search.

FIXED – General, RSS Discovery and Preview, Disability Access, Migration, Session Restore.

INCOMPLETE – Developers Tool, Tabbed Browser, Bookmarks and History.

INVALID – PDF Viewer, Panorama, Menus, Location Bar, Toolbar and Customization.

WONTFIX – Tabbed Browser, Bookmarks and History, PDF Viewer, Keyboard Navigation, Theme, Preferences.

WORKSFORME – Disability Access, Developers Tool, Build Config, Session Restore, Security, Location Bar, Untraiged.

Table 7: Relationship between component and last resolution for Trunk version

Component	DUPLICATE Reopen Rate	FIXED Reopen Rate	INCOMPLETE Reopen Rate	INVALID Reopen Rate	WONTFIX Reopen Rate	WORKSFORME Reopen Rate
Private Browsing	0.250	0.081	0.000	0.000	0.000	0.000
Developers Tool	0.222	0.147	0.250	0.097	0.178	0.274
File Handling	0.111	0.038	0.000	0.000	0.000	0.071
General	0.104	0.143	0.027	0.038	0.059	0.063
Build Config	0.100	0.071	0.000	0.000	0.000	0.143
Downloads Panel	0.091	0.023	0.000	0.000	0.250	0.000
Session Restore	0.080	0.106	0.000	0.000	0.000	0.114
Search	0.074	0.027	0.000	0.000	0.067	0.000
PDF Viewer	0.067	0.041	0.000	0.333	0.333	0.059
Panorama	0.061	0.048	0.000	0.167	0.250	0.050
Location Bar	0.060	0.028	0.000	0.100	0.000	0.083
Theme	0.051	0.059	0.000	0.048	0.120	0.048
Shell Integration	0.042	0.000	0.000	0.000	0.000	0.000
Keyboard Navigation	0.040	0.034	0.000	0.000	0.200	0.083
Security	0.037	0.038	0.000	0.000	0.000	0.111
Toolbars and Customization	0.032	0.053	0.000	0.091	0.000	0.067
Tabbed Browser	0.030	0.043	0.286	0.077	0.091	0.075
Untriaged	0.030	0.000	0.000	0.000	0.000	0.063
Menus	0.022	0.040	0.000	0.154	0.059	0.063
Bookmarks and History	0.021	0.035	0.083	0.050	0.045	0.024
Preferences	0.015	0.015	0.000	0.000	0.056	0.033
Disability Access	0.000	0.100	0.000	0.000	0.000	0.333
Migration	0.000	0.067	0.000	0.000	0.000	0.000
RSS Discovery and Preview	0.000	0.125	0.000	0.000	0.000	0.000

Table 8 depicts the relationship between component and last resolution field for the unspecified version. The results are given below which represents the relation between them:

DUPLICATE – Private Browsing, Download Panel, Developers Tool, Disability Access.

FIXED – File Handling, Tabbed Browser, Build Config, Session Restore, Menus.

INCOMPLETE – General, Location bar.

INVALID – Panorama, General.

WONTFIX – Keyboard Navigation, Social API, Toolbars and Customization, Preferences, Bookmarks and History.

WORKSFORME – Page Info Window, Theme, Disability Access, RSS Discovery and Preview, Developers tool, PDF viewer, Extension Compatibility, Location bar.

The results showed that which component had high probability to reopen based upon the last resolution field value.

Table 8: Relationship between component and last resolution for Unspecified version

Component	DUPLICATE Reopen Rate	fixed Rate	incomplete Rate	Invalid Rate	wontfix Rate	worksforme Rate
Download Panel	0.167	0.000	0.000	0.000	0.000	0.000
Developer Tools	0.148	0.083	0.000	0.027	0.011	0.137
Private browsing	0.125	0.105	0.000	0.000	0.000	0.000
Disability Access	0.105	0.091	0.000	0.000	0.000	0.167
Social API	0.100	0.038	0.000	0.000	0.200	0.000
Session restore	0.067	0.132	0.000	0.056	0.091	0.080
Panorama	0.056	0.057	0.000	0.111	0.077	0.111
GENERAL	0.055	0.091	0.147	0.089	0.179	0.085
Search	0.050	0.118	0.000	0.071	0.000	0.077
Keyboard Navigation	0.049	0.125	0.083	0.063	0.250	0.105
Tabbed Browser	0.043	0.267	0.045	0.069	0.000	0.067
Extension Compatibility	0.038	0.000	0.000	0.000	0.000	0.045
Location Bar	0.037	0.083	0.077	0.053	0.000	0.087
Toolbars and Customization	0.027	0.030	0.000	0.042	0.182	0.048
Theme	0.026	0.033	0.000	0.063	0.000	0.176
File Handling	0.026	1.000	0.000	0.000	0.000	0.077
Preferences	0.020	0.059	0.000	0.000	0.100	0.000
Menus	0.018	0.111	0.000	0.071	0.000	0.091
Bookmarks and History	0.017	0.000	0.000	0.000	0.067	0.000
Security	0.014	0.000	0.000	0.000	0.000	0.000
Shell Integration	0.014	0.000	0.000	0.000	0.000	0.000
Build Config	0.000	0.231	0.000	0.000	0.000	0.000
PDF Viewer	0.000	0.054	0.000	0.000	0.000	0.100
Sync	0.000	0.031	0.000	0.000	0.000	0.000
Page Info Window	0.000	0.000	0.000	0.000	0.000	0.333
RSS Discovery and Preview	0.000	0.000	0.000	0.000	0.000	0.143

Evaluating the Accuracy of Model

To check the accuracy of model, re-open precision is evaluated of components, component and severity, component and last resolution. To calculate re-open precision [2], TP (True Positive) and FP (False Positive) values are obtained. A parameter is said to be TP if its reopen rate is equal to or more than 0.05 in both the versions Trunk and Unspecified. And a parameter is said to be FP if its reopen rate is greater than or equal to 0.05 in one version and less than 0.05 in other version. Then the Re-open Precision can be calculated as given below:

$$\text{P(re-open)} = \frac{TP}{TP+FP}.$$

A precision value of 100 % would indicate that every bug we classified as re-opened was actually re-opened.

To estimate the accuracy of this model, firstly we calculated the re-open precision of Trunk version over unspecified version based upon components, component and severity, component and last resolution. And then re-open precision of unspecified over Trunk version as shown in the table 9.

Table 9: Re-open precision

Parameter		Re-open Precision P(re-open) Trunk over Unspecified	Re-open Precision P(re-open) Unspecified over Trunk
Based upon component		.83	.77
Based upon component and severity	High severity	.57	.50
	Medium severity	.64	.69
	Low severity	.67	.86
Based upon component and last resolution	DUPLICATE	.58	.78
	FIXED	.60	.40
	INCOMPLETE	0	0
	INVALID	.50	.44
	WONTFIX	.36	.50
	WORKSFORME	.70	.60

It is observed that the reopen precision for the components is calculated to be 83% and 77%, which is a high percentage of accuracy. It indicates that the prediction model which is described has accurately predicted the reopen bugs. Also the re-open precision for different severity levels is quite high. In case of last resolution, the re-open precision for DUPLICATE, FIXED and WORKSFORME is accurately predicted whereas for iNCOMPLETE, INVALID and WONTFIX the re-open precision is low.

IV CONCLUSIONS AND DISCUSSIONS

In this study, it is determined that which factors indicate whether a bug will be re-opened or not. Knowing which factors are attributed to re-opened bugs prepares practitioners to think twice before closing a bug. In this study, Bugzilla database for Mozilla Firefox is evaluated. Then extraction of reopened bug reports from Bugzilla database for Mozilla Firefox is done. Then the manual examination of the reopened bug report is performed. When the required fields are extracted from the bug reports, analysis of data is performed to characterize which factors influence the bug reopen rate.

When the re-open rates for the components, different severity levels and the last resolution is calculated for the trunk version. Then the analysis indicates that the components Developers Tool, General, Session Restore, Disability Access, Private Browsing, Build Config, Panorama, PDF Viewer has high re-open rate. Also the bugs with High severity have more chances of reopen as compared to Medium and Low severity. Similarly the re-open rate of bugs with the last resolution as DUPLICATE, FIXED and WORKSFORME is high.

And when analysis is performed on the unspecified version, then significant results are obtained as for the trunk version. Shihab et al. [2] who predicted reopened bugs in the Eclipse project. They used measures from four dimensions—work habits, bug report, bug fix and team—as input for decision trees (C4.5), which predicted reopened bugs with a precision of 62.9% and a recall of 84.5%. Whereas Zimmermann et al. [1] characterized the bug reopen process using a mixed methods approach: they qualitatively identified causes for bug reopens based on the survey responses of Microsoft engineers and performed a quantitative analysis using bug reports from the Windows operating system to assess the impact of the various factors.

But in this study, a relationship between factors is also drawn which can accurately predict bug reopen probability. It can be predicted that a component with specific severity level and last resolution has more probability to be re-opened for both versions. The re-open precision for the components is calculated to be 83% and 77% for the Trunk and Unspecified version respectively, which is a significant percentage of accuracy. It indicates that the prediction model which is described has accurately predicted the reopen bugs. Also the re-open precision for different severity levels is quite high. In case of last resolution, the re-open precision for DUPLICATE, FIXED and WORKSFORME is accurate predicted whereas for INCOMPLETE, INVALID and WONTFIX the re-open precision is low. The findings of this work contribute towards better understanding of what factors impact bug re-openings so they can be examined more carefully. Doing so will reduce the number of re-opened bugs and the maintenance costs associated with them.

REFERENCES

[1] Zimmermann T, Nagappan N, Guo PJ, Murphy B (2012) "Characterizing and predicting which bugs get reopened". In: ICSE '12: proceedings of the 34th international conference on software engineering, pp 495–504.

[2] Shihab, E., Ihara, A., Kamei, Y., Ibrahim, W.M., Ohira, M., Adams, B., Hassan, A.E., and Matsumoto, K. i. "Predicting Re-opened Bugs: A Case Study on the Eclipse Project". In Proceedings of the 17th Working Conference on Reverse Engineering (2010), 249-258.

[3] Hooimeijer, P. and Weimer, W. "Modeling bug report quality". In Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (2007), 34-43.

[4] Panjer, L.D. "Predicting Eclipse Bug Lifetimes". In MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories (2007).

[5] Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., and Zimmermann, T."What Makes a Good Bug Report?" In FSE '08: Proceedings of the 16th International Symposium on Foundations of Software Engineering (November 2008).

[6] Anbalagan, P. and Vouk, M. "On Predicting the Time taken to Correct Bugs in Open Source Projects" (short paper). In ICSM '09: Proceedings of the International Conference on Software Maintenance (September 2009).

[7] Mockus, A., Fielding, R.T., and Herbsleb, J.D."Two case studies of open source software development: Apache and Mozilla". ACM Trans. Softw. Eng. Methodol., 11 (2002), 309-346.

[8] Herbsleb, J.D. and Mockus, A." An Empirical Study of Speed and Communication in Globally Distributed Software Development" IEEE Trans. Software Eng., 29 (2003), 481-494.

[9] Cataldo, M., Herbsleb, J.D., and Carley, K.M. "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity". In ESEM '08: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement (2008), ACM, 2--11.

[10] Aranda, J. and Venolia, G. "The Secret Life of Bugs: Going Past the Errors and Omissions in Software Repositories". In ICSE' 09: Proceedings of the 31st International Conference on Software Engineering (2009).

[11] Breu, S., Premraj, R., Sillito, J., and Zimmermann, T. "Investigating Information Needs to Improve Coo"peration Between Developers and Bug Reporters". In CSCW '10: Proceedings of the ACM Conference on Computer Supported Cooperative Work (February 2010).

[12] Bettenburg, N., Premraj, R., Zimmermann, T., and Kim, S. "Duplicate Bug Reports Considered Harmful. Really ?" In ICSM '08: Proceedings of the 24th IEEE International Conference on Software Maintenance (September 2008), 337--345.

[13] Jeong, G., Kim, S., and Zimmermann, T. "Improving Bug Triage with Bug Tossing Graphs". In ESEC-FSE '09: Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on Foundations of Software Engineering (2009).

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

[15] E. Giger, M. Pinzger, and H. Gall. "Predicting the fix time of bugs". In RSSE, 2010.

[16] Ko, A.J., Myers, B.A., and Chau, D.H."A Linguistic Analysis of How People Describe Software Problems". In VL/HCC '06: Proceedings of the 2006 IEEE Symposium on Visual Languages and Human Centric Computing (2006), 127-134.

[17] Bertram, D., Voida, A., Greenberg, S., and Walker, R. "Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams". In CSCW '10: Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (2010), 291-300.

[18] Ko, A.J. and Chilana, P.K. "How power users help and hinder open bug reporting". In CHI '10: Proceedings of the 28th International Conference on Human Factors in Computing Systems (2010), 1665-1674.

[19] Anvik, J., Hiew, L., and Murphy, G.C. "Who should fix this bug?" In ICSE '06: Proceedings of the 28th International Conference on Software Engineering (2006), 361--370.

[20] Anvik, J. and Murphy, G. "Reducing the Effort of Bug Report Triage: Recommenders for Development-oriented Decisions". ACM Transactions on Software Engineering and Methodology (TOSEM).

[21] Canfora, G. and Cerulo, L. "Supporting change request assignment in open source development"t. In SAC '06: Proceedings of the 2006 ACM Symposium on Applied Computing (2006), 1767--1772.

[22] Canfora, G. and Cerulo, L. "Fine grained indexing of software repositories to support impact analysis". In MSR '06: Proceedings of the International Workshop on Mining Software Repositories (2006), 105-111.

[23] Hiew, L. "Detection of duplicate bug reports" , 2006. The University of British Columbia.

[24] Runeson, P., Alexandersson, M., and Nyholm, O. "Detection of Duplicate Defect Reports Using Natural Language Processing". In ICSE '07: Proceedings of the 29th International Conference on Software Engineering (2007), 499--510.

[25] Jalbert, N. and Weimer, W. "Automated duplicate detection for bug tracking systems". In DSN '08: Proceedings of the Annual IEEE/IFIP International Conference on Dependable Systems and Networks (2008), 52-61.

[26] Wang, X., Zhang, L., Xie, T., Anvik, J., and Sun, J. "An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information". In ICSE '08: Proceedings of the 30th International Conference on Software Engineering (May 2008).

[27] Menzies, T. and Marcus, A. "Automated severity assessment of software defect reports". In ICSM '08: Proceedings of the 24th IEEE International Conference on Software Maintenance (September 2008), 346-355.

[28] Bugzilla Documentation – Life Cycle of a Bug. Available online at http://www.bugzilla.org/docs/tip/html/lifecycle.html.

[29] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" in MSR '05: Proceedings of the 2005 international workshop on Mining software repositories, 2005, pp. 1-5.

[30] "Bugzilla for Mozilla", 2013. http://bugzilla.mozilla.org.

[31] http//Mozilla.org/en-US/firefox.

[32] Mittal, P., Singh, S., and Kahlon, K.S., 2011. "Identification of Error Prone Classes for Fault Prediction Using Object Oriented Metrics", ACC 2011, Part II, Communications in Computer and Information Science, Vol. 191, pp. 58-68.