IDENTIFICATION AND ANALYSIS OF CAUSES FOR SOFTWARE BUG REJECTION WITH THEIR IMPACT OVER TESTING EFFICIENCY

Ghazia Zaineb[1] and Dr. Irfan Anjum Manarvi[2]

[1]Department of Engineering Management, Center for Advanced Studies in Engineering, Islamabad, Pakistan
ghazia.zaineb@gmail.com
[2]Department of Engineering Management, Center for Advanced Studies in Engineering, Islamabad, Pakistan
irfanmanarvi@yahoo.com

ABSTRACT

A significant percentage of software bugs reported during test cycles falls in a category of Invalid bug reports also known as rejected bugs. This research presents the actual percentage of bugs rejection based on data collected from bug tracking system. This paper provides a list of reasons behind bug rejection, their relation with severity level and possible threats that can affect software testing efficiency with reference to the life of a rejected bug.

KEYWORDS

Invalid Bug Report, Rejected Bug, Software Testing, Bug Severity, Testing Efficiency

1. INTRODUCTION

Software Testing is an important and most critical phase in Software Development Life Cycle (SDLC). The major activity performed during software testing is to identify bugs and according to [3], more than 40% of project time is consumed over this activity. Bugs identification is usually performed by testing team and followed by a team review to priorities fixes. In [1], number of defects is also a way to determine the quality of software. Every bug has a severity level and a status associated with it. [2] Defines severity as *effort* required to fix a particular bug. Status of bug is helpful in keeping the track of progress over reported bug. Out of a common list of bug statuses one status of concern in this research is rejected bug. Bug status could be new, open, fixed, re-opened, verified, rejected and close. In current case, the testing team marks bug as closed after accepting the reason of bug rejection.

In all reported bugs there is a possibility that a particular bug may not worth fix. There can be multiple reasons behind not providing a fix but in common, all such bugs are marked as "rejected" status in bug tracking tool. In our research, we tried to find out as many causes as possible that can lead to bug rejection. One reason reported in [4] is the quality of information provided in bug report. By quality they mean the level of information that was required by the development team and what actually was provided by the bug reporting team. Other possible reasons provide in [4] are system configuration, language differences, duplicate information and incomplete data regarding the reported bug. [4] Identifies 21 problems in total out of which *steps to reproduce* got the highest weight (79%). A duplicate bug report is also a very common issue and in some projects one quarter of reported bugs is duplicate [9]. Identification of duplicate bugs from a list of already reported bugs is itself a very time consuming activity [9]. A lot of work has been done to propose tools and techniques for avoiding bug duplication in [10] and [11].

Another common reason behind an invalid bugs report is incomplete documentation. According to [5], documents are essential source of information for both bug detection and resolution. The two most important documents for software development are software requirement specification and design document. It is important that both the development and testing team share same version of such documents to avoid ambiguity and invalid bug report. A study conducted over a project shows that rejection rate remains stable over the number of years for that project in which out of five identified causes; two major reasons for rejection are misunderstood functionality (27.73%) and wrong sequence of steps provided in test case (45.68%) [8]. Wrong steps to reproduce a bug can also lead it to rejection [4]. It is also possible that the reported bug occurred in a particular environment [12]. So if there is a difference in test environment then there exist fair chances for the bug that it will not be reproduced. Quality of bug report contents is very important for understanding the exact bug scenario. [13] Presents a model based on contents of bug report to identify the cost of fix. Importance of information required for fixing a bug has been discussed in [15] as well. Poor bug report quality is also discussed in [14] as a factor of reassignment. Incomplete information makes a bug fix more difficult.

It is important to identify and remove causes that lead to bug rejection to eliminate extra burden over the software project in terms of cost associated with the reported bug. [6] Reports approximately 20 minutes consumed over analysis and validation of a bug. So if the same time is consumed over a duplicate bug then it means a direct negative impact over software *maintenance* [6]. Bug identification is an expensive activity [7] but it is essential to avoid possible ensuing damage. The current research also presents similar facts of bug cost depending on the time it takes for identifying bug and marking it close after review and rejection.

Other than an impact over project cost, bug rejection also effects testing efficiency. One important testing matrix is the defect rejection rate [1]. Higher the rejection rate means lower the testing efficiency. Also there is a possibility that much of testing effort was consumed over identification of such bugs that can be utilized in identification of valid bugs. Software testing efficiency is also discussed in terms of reported bugs, time, effort, cost and number of test case execution in [9] and [16].

In our research we have identified the significance of number of rejected bugs in software projects and then presented the facts over number of invalid reports that falls in identified categories of causes for bug rejection. Our research provides a statistical analysis over the relationship among number of reported bugs and rejected bugs. The results of this research also provide correlation analysis of reason of rejection with bug severity. We concluded this research by identifying the most frequent reasons for bug rejections, their impact over testing efficiency and recommendations to decrease the rejection rate.

1.1. Objective

The objective behind this research is to bring attention towards the increasing rate of software bug rejections that effects testing efficiency in terms of number of bugs identified and time consumed in bug identification and reporting. The research brings statistical facts about invalid bug reports that helps in estimating cost of effort spent over rejected bug based on duration in number of days between the bug reported and marked as closed after acceptance of rejection by the reporting team. The results of this research highlight the main causes of bug rejection which should be avoided to decrease the overhead by removing the number of rejected bugs.

1.2. Research Scope and Limitations

This research covered facts regarding software defects only that are related to functionality, GUI and performance. In some organization document defects like problem in user guide and missing information in SRS / design documents are also reported in defect tracking system but such bugs were not under scope of current research. Also the selected projects were not considered for their type i.e. web/desktop/mobile application because of the data availability limitation. Also we are at an assumption that project type will not affect the research findings.

1.3. Research Questions

This research provides answers to a list of following questions:

1. What is the percentage of rejected software bugs?
2. What kind of correlation exists between rejected bugs and total identified bugs?
3. What is the significance of bug rejection rate over testing matrixes?
4. What are the major causes behind defect / bug rejection?
5. Is there any relationship between bug rejection and their severity level?
6. How to eliminate the possibility of invalid bug reports?

2. MATERIAL AND METHOD

This research is conducted over date collected from bugs tracking tool shared by developers and testers for keeping record of software bugs reported during multiple cycles of software testing. A total of 17 projects with their separate bug's repositories were considered for collecting date to identify bug rejection matrix. Only those bugs were considered that have been either marked as closed after being rejected or were in rejected status. Deferred bugs or issues marked for future releases were not considered as rejected bugs.

Out of all 17 projects we have considered the most recent five to collect a sample to 200 rejected bugs in order to identify rejections causes. All bugs were collected in sequence so that maximum possible causes of bug rejections can be identified in a single software project. Data collection was done independent of the bug severity level and comments provided for rejection. Each bug was then analysed over rejection comments and was classified manually in proposed cause's category for further analysis.

Variables used for statistical analysis in this research were Project ID, Project Name, Total Defects, Closed Defects, Rejected Defects, Bug ID, Bug Severity, Log Date, Closed Date, and Rejection comments. Further a list of computed variable used were Total Number of Rejected bugs, Bug duration and reason category. All variables were treated as continuous except the two categorical variables of bug severity and rejection reason which were analysed as nominal variables.

Statistical Package for Social Sciences (SPSS 16.0) was used for statistical analysis of collected data. In order to find correlation between two variables we have used Pearson's coefficient of correlation with one tailed test of significance.

3. FACTS AND FINDINGS

Table 1 presents stats of bug rejections for all 17 projects. The collected data provides a cumulative rejection rate of 14.40% with a total of 1359 rejections out of a total 9434 reported bugs.

Table 1. Bug rejection statistics.

Project ID	Total Bugs	Rejected Bugs	Rejection Ratio
1	428	46	10.75%
2	578	89	15.40%
3	188	22	11.70%
4	361	44	12.19%
5	2161	298	13.79%

6	35	0	0
7	113	18	15.93%
8	2142	392	18.30%
9	239	17	7.11%
10	1043	154	14.77%
11	171	8	4.68%
12	109	11	10.09%
13	1214	163	13.43%
14	470	75	15.96%
15	82	17	20.73%
16	69	5	7.25%
17	31	0	0

Rejection rate is not constant among all projects. From Table 2 the average rejection ratio is 11.30% which is quite significant for considering it as a percentage of overhead for the testing schedule in terms of time and cost.

Table 2. Rejection ratio statistics.

No of Valid Values	17
Missing Values	0
Mean	11.2980
Median	12.1884
Std. Deviation	5.89764

Figure 1 shows scatter plot for correlation of total number of bugs and total number of rejected bugs. The scatter plot shows a strong positive linear relationship between the two counts. Value of R2 is 0.989 which means the correlation is at 99.99% confidence interval. With this value of correlation it is obvious the increased number of reported bugs will also increase the number of rejected bugs.

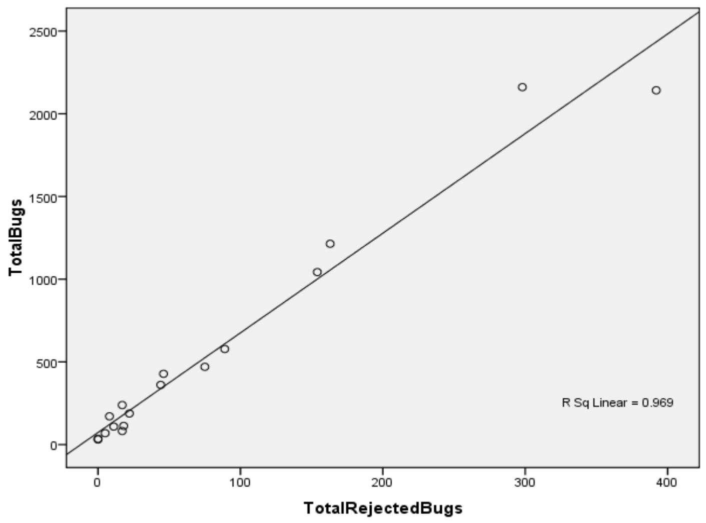

Figure 1. Scatter plot for correlation

Table 3 shows the facts generated by SPSS for correlation analysis of the two scalar type variables where N shows the number of projects considered for obtaining the values for the two variables.

Table 3. Correlation analysis.

Correlation			
		TotalRejectedBugs	TotalBugs
Pearson Correlation	TotalRejectedBugs	1.000	.984
	TotalBugs	.984	1.000
Sig. (1-tailed)	TotalRejectedBugs	.	.000
	TotalBugs	.000	.
N	TotalRejectedBugs	17	17
	TotalBugs	17	17

Table 4, 5 and 6 presents data generated for regression analysis of the two variables. Value of adjusted R Square presents that 96.9% variation in number of rejected bug reports is due to the increase or decrease in total number of reported bugs. The regression equation will be helpful in predicting the number of rejection based on reported bugs. The regression equation drawn from Table 6 is given below:

Total Number of Rejected Bugs = - 9.339 + 0.161(Total number of reported bugs)

Table 4. Regression analysis model summary.

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.984[a]	.969	.967	20.431
a. Predictors: (Constant), TotalBugs				

Table 5. Regression analysis ANOVA table.

ANOVA[b]						
Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	197045.752	1	197045.752	472.065	.000[a]
	Residual	6261.189	15	417.413		
	Total	203306.941	16			
a. Predictors: (Constant), TotalBugs						
b. Dependent Variable: TotalRejectedBugs						

Table 6. Regression analysis coefficients table.

Coefficients[a]								
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Collinearity Statistics	
		B	Std. Error	Beta			Tolerance	VIF
1	(Constant)	-9.339	6.437		-1.451	.167		
	TotalBugs	.161	.007	.984	21.727	.000	1.000	1.000
a. Dependent Variable: TotalRejectedBugs								

Above all was the statistical analysis of bug rejections at project level. Figure 2 presents a Bar Chart for causes identified for bug's rejection based on rejection comments collected from a total of 200 rejected bugs. A total of 19 reasons have been identified which have been explained in section 4 of this research paper.

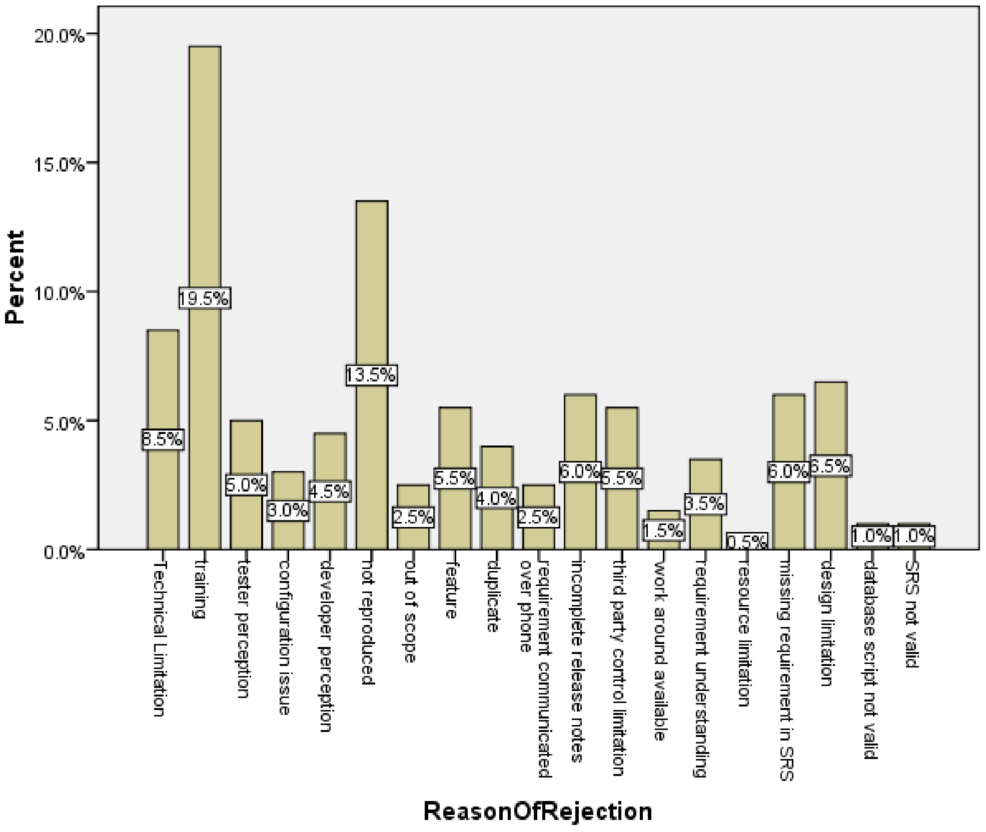

Figure 2. Bar Chart for Reasons of Bug Rejection

Table 7 is generated with help of SPSS software for descriptive statistics of rejection reasons variable. The date in the table is arranged in descending order to keep a view of Pareto analysis. The cumulative percentage shows that 41.5% of bug rejections are because of training over the test release, not reproduced due to incomplete steps reported in bug report and technical limitations. Out of all collected comments only one case turned out to be resource limitation as a cause of bug rejection.

Table 7. Reason of rejection statistics.

ReasonOfRejection					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	training	39	19.5	19.5	19.5
	not reproduced	27	13.5	13.5	33
	Technical Limitation	17	8.5	8.5	41.5
	tester perception	10	5.0	5.0	46.5
	design limitation	13	6.5	6.5	53
	missing requirement in SRS	12	6.0	6.0	59
	incomplete release notes	12	6.0	6.0	65
	feature	11	5.5	5.5	70.5
	third party control limitation	11	5.5	5.5	76
	developer perception	9	4.5	4.5	80.5
	duplicate	8	4.0	4.0	84.5
	requirement understanding	7	3.5	3.5	88.0
	configuration issue	6	3.0	3.0	91
	out of scope	5	2.5	2.5	93.5
	requirement communicated over phone	5	2.5	2.5	96
	work around available	3	1.5	1.5	97.5
	database script not valid	2	1.0	1.0	98.5
	SRS not valid	2	1.0	1.0	99.5
	resource limitation	1	.5	.5	100
	Total	200	100.0	100.0	

Table 8 shows the frequency of severity level found in rejected bugs. 63% rejections are found to be for functional bugs. P0 status shows the severity of exceptions occurred in application at various test stages. Only 10% rejected bugs have severity p0. P2 shows the severity of GUI bugs that are 26% and only 6% rejection have severity level related to enhancements and suggestions represented by e1 and e2 respectively.

Table 8. Bug severity statistics.

BugSeverity					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	p1	126	63.0	63.0	63.0
	p2	52	26.0	26.0	89.0
	p0	10	5.0	5.0	94.0
	e1	10	5.0	5.0	99.0
	e2	2	1.0	1.0	100.0
	Total	200	100.0	100.0	

In order to answer one of the current research questions regarding relationship between rejection cause and bug severity we have developed a matrix scatter plot for the three variables including rejection reason, severity and duration of rejected bug.

Figure 3 shows the matrix scatter plot from which it is clear that there is no significant relationship exists between the reason of rejection and bug severity. SPSS provides the value of R Square as 0.02 for the linear relationship between these two variables. It means that a reported bug can not be rejected only because of its severity. For example, the severity 'e1' and 'e2' were used for enhancements only and therefore there are fair chances of such bugs with severity e1 or e2 to get rejected due to time or resource limitation but still this severity level is not reflecting any correlation with rejection. One observed reason for this issue is that such bugs are usually marked as deferred if can not be fixed in current release.

Figure 3 also shows the insignificant relationship between rejection reason and bug duration. For example a duplicate bug can be identified on reported date or may take a number of days depending on the duration of bug reviews. From the matrix scatter plot one other inference is about the relationship of time consumed over the rejected bug and its severity or rejection reason. Here again no significant correlation exists between the variables and it is obvious that duration between the bug report time and its closing / rejection depends on how long would it take for the team to come up with an agreed point of view. Every rejection follows a sequence of reviews and discussion between the development and test team. During such reviews a rejected bug gets re-opened multiple times until the reason of rejection proves to be satisfactory for the reporting team or tester to accept rejection and finally close down the issue.

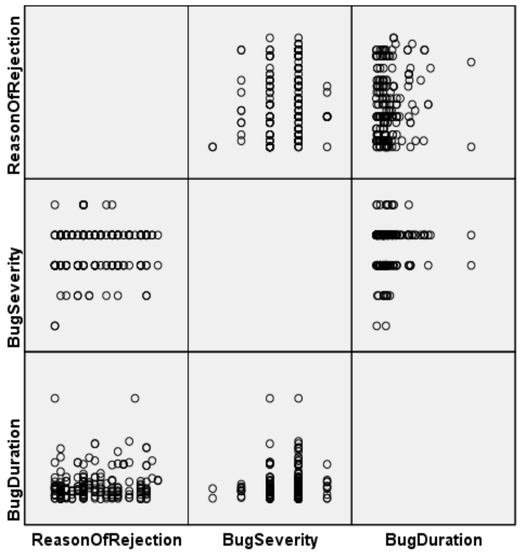

Figure 3. Matrix scatter plot

4. CAUSES OF BUG REJECTION

Based on literature, collected data and personal experience we developed a cause – affect diagram for all possible causes that results in invalid bugs report or rejected bug. Figure 4 presents this cause – affect diagram which indicates five major sources of causes for bug rejection. A total of 19 causes identified during data analysis have been arranged in sub causes of the main problem area.

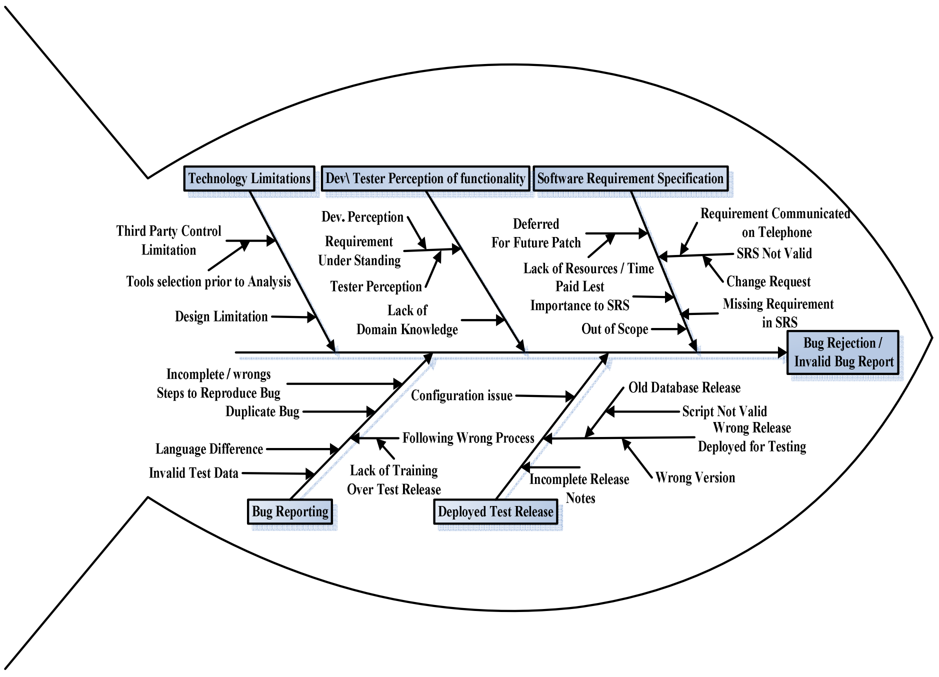
Technology Limitations
Dev\ Tester Perception of functionality
Software Requirement Specification
Third Party Control Limitation
Tools selection prior to Analysis
Design Limitation
Dev. Perception
Requirement Under Standing
Tester Perception
Lack of Domain Knowledge
Deferred For Future Patch
Lack of Resources / Time Paid Lest Importance to SRS
Out of Scope
Requirement Communicated on Telephone
SRS Not Valid
Change Request
Missing Requirement in SRS
Bug Rejection / Invalid Bug Report
Incomplete / wrongs Steps to Reproduce Bug
Duplicate Bug
Language Difference
Invalid Test Data
Configuration issue
Following Wrong Process
Lack of Training Over Test Release
Old Database Release
Script Not Valid
Wrong Release Deployed for Testing
Incomplete Release Notes
Wrong Version
Bug Reporting
Deployed Test Release

Figure 4. Cause-affect diagram for rejected bugs

Following is the list of major causes of bug rejection with their brief description including sub causes:

Software requirement specification (SRS): SRS could result in bug rejection if a certain requirement is not given in SRS document or the reported bug demands a functionality that is not in scope of SRS. Missing requirements could be a result of direct communication with client over phone and therefore may skip from SRS. Bug report regarding already deferred requirement due to resource or time limitation may also face rejection if the bug reporter is not aware of the deferred functionality or the provided SRS document is out dated.

Developer or Tester perception of application functionality: Another cause of bug rejection is about how the development and testing team perceived the requirements. Both the developer and tester have their own point of view and therefore an issue considered as bug for testing team could be an application feature for the development team. Lack of domain knowledge adds a lot in such differences of requirement understanding.

Technology Limitation: most of the time developers use existing modules to fit in new requirements with a little or no customization. Such tools or modules are helpful in saving time and effort but actually most of the third party controls do not allow full customization and therefore bug reports related to such tools are rejected on the ground of technical limitation. Also certain functionalities available in web modules cannot be provided in desktop application. Such issues are also part of technology limitation. It is also possible the fix of a certain bug report is not supported by application's current design and fixing one bug could lead to a critical design change therefore a bug will be rejected as far as it workarounds are available and can save design change.

Deployed Release: The most critical source of bug rejections identified from the collected data is the deployed release itself. Test release deployed for testing usually lacks detailed release notes. Incomplete release notes results in bug reports that are already known issues at development end therefore they simple reject the reported bug. In some cases the database script provided with the test release was not valid. The deployment team usually gets the latest version of application only and uses the same old database script. In such situation when a bug is reported the cause of bug is actually the old database script which does not need a fix but just a replacement of file therefore the reported bug is rejected. There are chances that some of the configuration settings are not same at development and testing server so a bug generated due to configuration issue will be rejected and does not require any development effort for the fix.

Bug Reporting: The most common source of bug rejection is the bug report itself. If the bug report is not complete or has missing steps then it is impossible for the developers to identify the actual cause of bug and therefore the bug turn to be invalid. Language differences between the bug reporter and the person who will fix the bug can generate more difficulties in identifying the actual cause behind bug. Such bugs remain not reproduced for developers and sometimes for testers as well if the person who actually reported that particular bug is no more working over the same project. There is possibility that the required functionality reported in bug do exists in application but the testers are not informed properly due to lack of training provided to them over the deployed test release. Testers who report a bug by following wrong sequence of steps gets the bug rejected after receiving correct sequence from the developers after bug review.

5. IMPACT OVER TESTING EFFICIENCY

A number of software testing matrixes are based on total number of defects/ bugs reported therefore it is a common practice in developing software matrixes to eliminate the number of rejected bugs from the count to keep project performance realistic. For instance to calculate bug fixing efficiency the number of fixed bugs are divided by total number of reported bugs. Now if rejected bugs are also included in total number of bugs then definitely the bug fixing ratio will be decreased. The decrease in efficiency impacts negatively over software project performance.

So to avoid this performance decrease usually the same negative impact is transferred to testing by eliminating the number of rejected bugs that actually decreases the total number of identified bugs per unit time.

On important testing matrix is the rejection rate calculated as follows:

Bug Rejection Rate = (No. of Invalid bug reports / total number of bug reports)*100

Higher rejection rate shows low testing efficiency. Except this matrix, remaining all testing matrixes do not consider the rejection count but in actual the time consumed over rejected bugs is included in total testing effort while measuring the overall test efficiency and effectiveness. That means if execution of a certain test case resulted in 4 bugs out of which 3 have been rejected then test effectiveness will be decreased in terms of average number of bugs per test case where as the overall schedule shows the entire testing effort as consumed over identification of just one bug. Due to limited time available for testing effort it is also possible that some important test cases could not be executed because of the time spent over identification of invalid reports.

A software bug with status "not reproduced" takes more than double time in verification as compared to one marked as fixed. Such rejection results in a lot of rework for both the testing and development team that is, if the bug is not reproduced due to test and development environment differences then development has to identify the actual environment settings and if the bug is not reproduced due to wrong sequence of steps then testing team has to put effort again for identification of correct sequence. During this activity the bug remains open for a large number of days. On average this duration is approximately 22day as calculated from the collected data for rejected bugs during the research. In worse case if a certain functionality is not working at testing machine just because of some configuration issue or missing column in data base script then there are fair chances that other test cases dependent on the results of this scenario may not be executed or can generate more bug reports which will finally increase the count for rejected bugs.

6. CONCLUSIONS

The major problem areas causing bug rejections are bug reports and insufficient knowledge of tester over the developed software. Instead of wasting time over reviews and rework it would be better to provide training to the test team before deploying the test release. Also the testers should be careful while recording steps. They should provide the sequence in such a way that any other person can easily reproduce the same scenario even in their absence. Testing and development members should be a part of requirement gathering team for better understanding of application requirements. It will help the project team to have a clear idea of scope and limitations to avoid rework. Keeping precise and updated documentation also helps in avoiding rework for already know issues and strong domain knowledge will benefit in identifying technical limitations at early stage of software development. Taking all these points in consideration can increase the testing efficiency and effectives.

All potential causes behind bug rejection indicated in this research will help in eliminating possibility of invalid bug reports. The research presents an average rate of approximately 12% for software bug rejections for a total of 19 causes of rejections. This rate affects all testing matrixes negatively and adds a lot in hidden cost of software projects in terms of time consumed over testing effort. Eliminating the rejected bugs could not save the progress discrepancies of overall project so it is better to pay attention towards eliminating the causes of such a high bug rejection rate.

ACKNOWLEDGEMENTS

Support for statistical analysis was provided by Mr. Wasim Bhatti working as research associate and PhD scholar at Center for Advanced Studies in Engineering, Islamabad.

REFERENCES

[1] Iacob, I.M., & Constanttinescu, R., (2008) "Testing: First Step towards Software Quality", *Journal of Applied Quantitative Methods*, Vol. 3, No. 3, pp241-253.

[2] Soner, S., Jain, A., Tripathi, A., & Litoriya, R., (2010) "A Novel Approach to calculate the severity and priority of bugs in software projects", *2nd international conference on education and technology and computer* (ICETC), Vol. 2, pp50-54.

[3] Trivedi, P., & Pachori, S., (2010) "Modelling and Analysis of Software Defects Prevention Using ODC", *International Journal of Advanced Computer Science and Applications* (IJACSA), Vol. 1, No. 3, pp75-77

[4] Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schroter, A., & Weiss, C., (2010) "What Makes a Good Bug Report?", IEEE *Transactions on Software Engineering*, Vol. 36, No. 5, pp618-643.

[5] Aranda, J., & Venolia, G., (2009) "The Secret Life of Bugs: Going Past the Error and Omissions in Software Repositories", *31st IEEE International Conference on software Engineering* (ICSE), pp298-308.

[6] Cavalcanti, Y.C., de Almeida, E.S., da Cunha, C.E.A., Lucrédio, D., & de Lemos Meira, S.R., (2010) "An Initial Study on the Bug Report Duplication Problem", *14th IEEE European Conference on Software Maintenance and Reengineering* (CSMR), pp264-267.

[7] Kumaresh, S., & Baskaran, R., (2010) "Defect Analysis and Prevention for Software Process Quality Improvement", *International Journal of Computer Applications*, Vol. 8, No. 7, pp42-47.

[8] Sun, J., (2011) "Why are Bug Reports Invalid?", IEEE *4th International Conference on Software Testing Verification and Validation*(ICST), pp407-410.

[9] Jalbert, N., & Weimer, W., (2008) "Automated Duplicate Detection for Bug Tracking Systems", IEEE *international Conference on Dependable Systems and Networks with FTCS and DCC*(DSN), pp52-61.

[10] Xiaoyin Wang, Lu Zhang, Tao Xie, Anvik, J., & Sun, J., (2008) "An approach to detecting duplicate bug reports using natural language and execution information", ACM/IEEE *30th International Conference on Software Engineering*(ICSE), pp461-470.

[11] Wang, D., Lin, M., Zhang, H., & Hu, H., (2010) "Detect Related Bugs from Source Code Using Bug Information", IEEE *34th Annual Computer Software and Applications Conference*(COMPSAC), pp228-237.

[12] Qin, F., Tucek, J., & Zhou, Y., (2005) "Treating Bugs As Allergies: A Safe Method for Surviving Software Failures", *Proceedings of the 10th conference on Hot Topics in Operating Systems*(HOTOS), Vol. 10, pp.19-19.

[13] Hooimeijer, P., & Weimer, W., (2007) "Modeling Bug Report Quality", *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*(ASE), pp34-43.

[14] Philip, J.G., Zimmermann, T., Nagappan, N., & Murphy, B., (2011) ""Not my bug!" and other reasons for software bug report reassignments", *Proceedings of the ACM 2011 conference on Computer supported cooperative work*(CSCW), pp395-404.

[15] Breu, S., Premraj, R., Sillito, J., & Zimmermann, T., (2010) "Information needs in bug reports: improving cooperation between developers and users", *Proceedings of the 2010 ACM conference on Computer supported cooperative work*(CSCW), pp301-310.

[16] Eldh, S., Hansson, H., Punnekkat, S., Pettersson, A., & Sundmark, D., (2006) "A Framework for Comparing Efficiency, Effectiveness and Applicability of Software Testing", *Techniques Proceedings Testing: Academic and Industrial Conference - Practice And Research Techniques* (TAIC PART), pp159-170.