A PROPOSED MODEL FOR IMPROVEMENT IN BUGS MANIFESTATION PROCESS

***Mahreen Shahid[1], Muhammad Haris Abid[2]**
[1]University of Agriculture, Faisalabad, Pakistan
[2]Shiblee College, Faisalabad, Pakistan
*Corresponding Author's email: mahreenshahid55@gmail.com

ABSTRACT: *Currently bugs manifestation process is being utilized in detection of different types of bugs. These bugs do not show up easily at development time and manifest themselves only as operational failure. There are many areas in bugs' manifestation process which can be improved or optimized. The purpose of this research is to analyze bugs manifestation process to find flaws. Two main processes areas are selected after analysis of bugs' manifestation process in the light of the flaws occurred. These processes are Development and Maintenance. In this research work I proposed a model to stop these bugs before and after development process to meet customer requirements. The suggested model will be efficient and more reliable compared to the existing one.*

Keywords: manifestation, processes, bugs, exploratory testing, regression testing, empirical, programming activity.

1. INTRODUCTION:

Bugs free software can only improve software quality and development processes. Scientists, crosswise over years, broke down bugs from various perspectives to enhance software quality and its processes. Many bugs efficiently cause a similar disappointment on a given stage (succession of) input(s). Alternately, there is a non-unimportant arrangement of bugs that cause a disappointment relying upon the condition of the execution of code, showing up as non-deterministic or transient, in which the disappointment does not happen unless the environment is in a certain state [4]. The method of delivering these systems is simplified in pre-release and post-release phases. The pre-release part of delivering a system includes the event and maintaining activities performed at Transaction of Software Engineering. The post-release part includes activities in test and production environments of the organization owning the systems [6]. Software applications are commonly built by integration legacy and outsider components. A generally known improvement regularly discovered in such frameworks is code maturing. [1]. A major drawback in bugs detection techniques is that they are not maintaining their performance once transferred from one system to a different. This lack of exchangeability is as a result of every software package tends to own specific options. These options stop from context factors together with application domain, development setting, language, development team etc [5]. Research and studies shows that the importance of characteristic the environment as security cause of bug publicity[2,4,7]. Memory [9], concurrency [8], and resource management [3] are also highlighting the execution environment of software bugs.

This research work includes the following sections: Sections 2 contain material and methods in terms of bugs manifestation process in Software Development Life Cycle (SDLC). Characteristics of Bug Manifestation Process discussed in Section. 3. Section 4 system model and methodology of Bugs Manifestation Process. This paper concluded by finding, conclusion and future work (Section 5, 6) however list of references followed at the end of this research work.

2. MATERIAL AND METHODS

The major concern of this research is to explain and explore bugs manifestation process in software development life cycle (SDLC). Bugs are encountered during verification activity and normalized also by programmers. Timely detection and correction of bug is objective to quality and reliability of software. In other case bugs may be converted into faults, errors or defects. Software cost can grow up to 100% if bugs transferred to customer at deployment phase. Therefore, we have explained current bug manifestation process and its pros and cons. Furthermore, we have included two testing techniques i.e. exploratory and regression to improve reliability and quality of software. Exploratory testing is used to find errors during development process and to stop operational failures. Regression testing is applied only if programmer finds bugs after development, regression testing helps to maintain the process. Additionally, both processes are empirically compared by applying them on some programming activity. Test cases are designed to explore the positive or negative impact of suggested approaches.

3. CHARACTERISTICS OF BUG MANIFESTATION PROCESS

The arrangement of starting conditions ("triggers") considered in the examination. Triggers are seen as the conditions essential for the bug to be established and increased up to the client interface as dissatisfaction. We first present a crucial framework model incorporating the considered triggers.

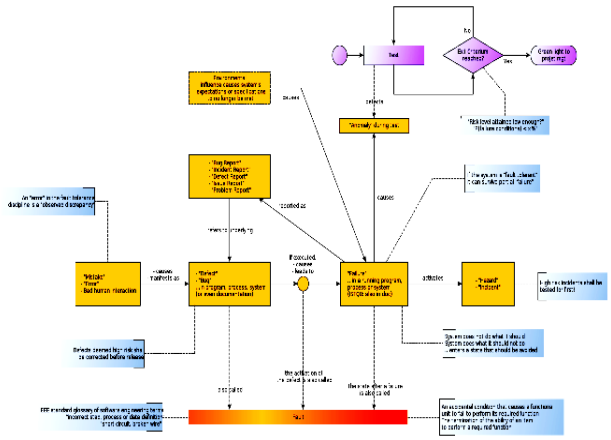

Figure 1: Bugs Manifestation Process

4. SYSTEM MODEL

We consider the application and the central external substances with a potential impact, specifically the customer and the execution environment. We expect an application as made out of strategies and/or strings talking with each other to accomplish the proposed limit, with correspondence channels executed by either an around the world (e.g., shared memory) or an area model (e.g., message exchange). The state of the application fuses the states of neighboring methodology (and/or strings), and of correspondence channels among them. An area state is the course of action of data (e.g., set away as variables in memory or reports) which the systems/strings can take a shot at (i.e., read from/create too).

After analyzing above programming activity we applied same activities by adopting a new model which contained exploratory and regression testing. We have designed some test cases to evaluate our proposed model. Furthermore, we have developed a regression model to justify our results empirically.

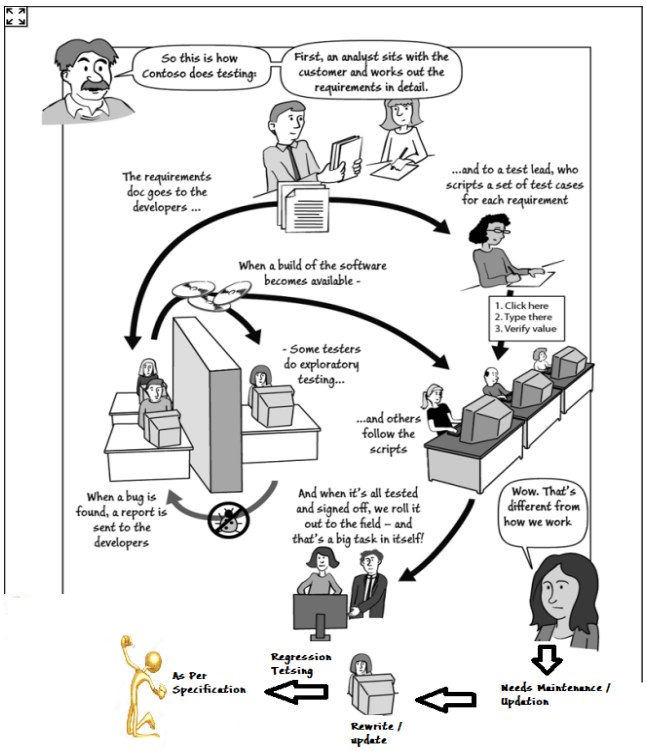

Figure 2: Proposed Model

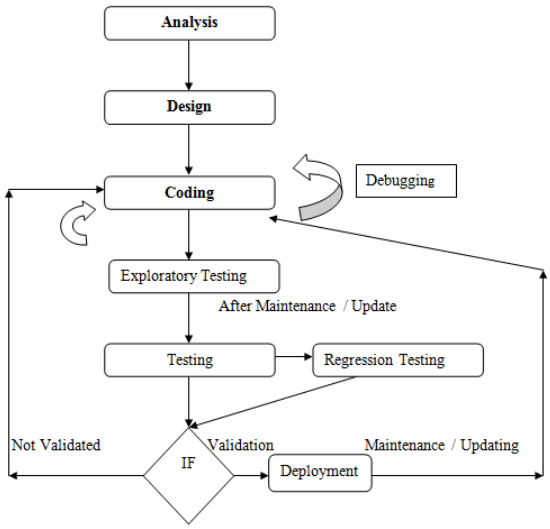

Figure 3: Flow Chart For Proposed Model

After deeply observing of the bug's manifestation process we have introduced two new phase in it. The first source of bugs is formal testing phase that includes exploratory testing during development phase, secondly, if bugs delivered to customer than regression testing will be applied at maintenance phase. Detecting bugs earlier leads not solely to low-cost and simplify, however, additionally early detection of bugs will play a significant role within the reliability of systems.

Empirically comparison is made on both processes (before including exploratory and regression testing and after including) by applying them on some programming activity. We developed hypothesis (H1: Inclusion of Exploratory and Regression testing makes Bugs manifestation process more efficient and reliable and HA: Inclusion of Exploratory and Regression testing does not make Bugs manifestation process more efficient and reliable). On the basis of Hypothesis results and conclusion is made. We also design a regression model to observe the impact of Exploratory and Regression testing on the reliability and efficiency of Bugs manifestation process. Regression Model is

$$Re = \alpha+\beta1\ (EXPT) + \beta2\ (RET) +\varepsilon$$

$$Ef = \alpha+\beta1\ (EXPT) + \beta2\ (RET) +\varepsilon$$

(Where Re is Reliability, Ef is Efficiency, EXPT is Exploratory Testing and RET is Regression Testing).

The suggested process and model will be efficient and more reliable in compare to existing one.

4.1 METHODOLOGY

Bugs manifestation process is used to explore bugs early in software development process (SDLC) to avoid time

consumption and improvement in the quality of the end product to satisfy customer. Impact of bugs grows if it is not detected during debugging or as early as the product is deployed or delivered to the customer. The intensity of bug becomes severe if it travels to end user. We have discussed bugs manifestation process and its current issues in chapter 3. In this chapter, firstly we presented bugs manifestation process's working result and its issues. Secondly, we presented results for exploratory and regression testing individually in SDLC. Last but not least we have presented results for our suggested model which we generated by including exploratory testing during traditional debugging phase and regression testing included after deployment when end user wants maintenance and Update in his / her product. At the end on the basis of results we have justified designed hypothesis i.e. (exploratory and regression testing improves software quality) by using regression equation.

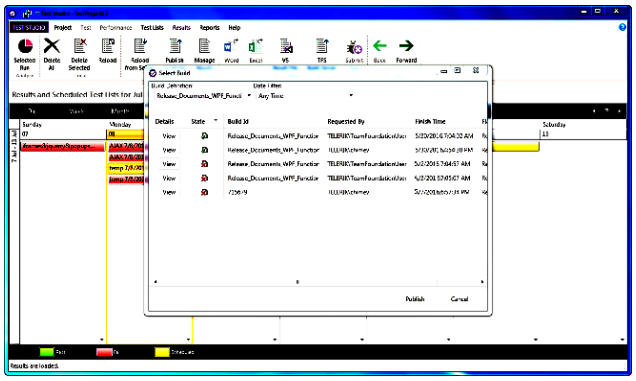

Figure 4: Adding Task In Test Studio For Exploratory Testing

Following figure represents the exploratory testing. As results are showing exploratory is 43% effective and in remaining cases it partially or tally failed.

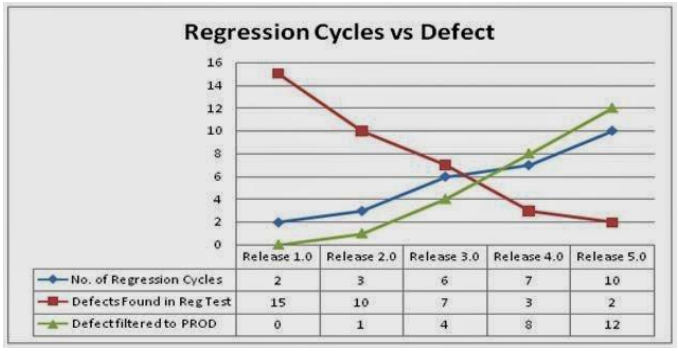

Figure 5: Test Results Of Exploratory Testing

Figure 7 shows that bugs / defects grow with a continuous maintenance/update of software. So we can say that regression testing is said to be partially passed.

5. RESULTS OF PROPOSED MODEL

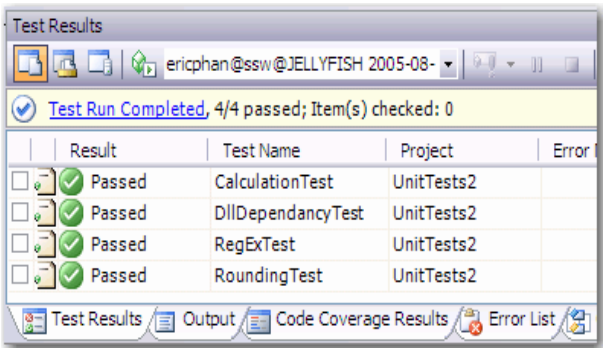

Figure 6: Regression Testing

than regression testing will be applied at maintenance phase. Detecting bugs earlier leads not solely to low-cost and

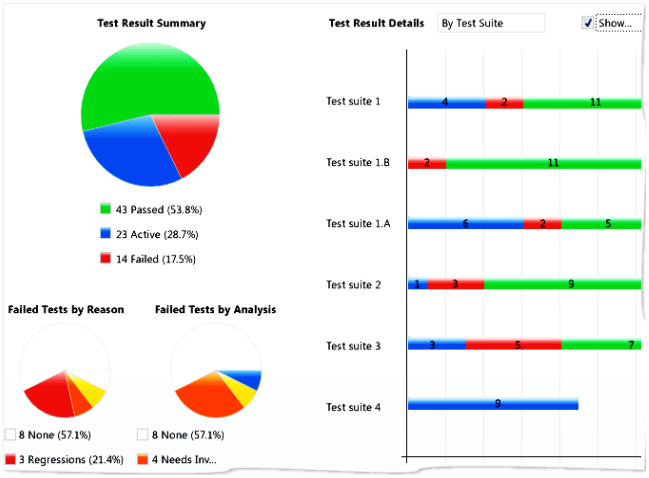

Figure 7: Regression Testing Results

simple, however, additionally early detection of bugs will play a significant role within the reliability of systems.

Following figure shows the results of proposed model. Gray Line shows the result for exploratory testing which means it is not performing well and required a lot of time. Similarly, Red line shows regression testing statistics which means it is also running in static behavior. The Blue line indicates results of our proposed model. The blue line shows as test cases increase the test results found less bugs and runs in average time. It means our proposed model is showing positive results. Hence we can say that with including exploratory and regression testing in SDLC bugs occurrence and manifestation can be reduced.

After a deep observation of bug's manifestation process we have introduced two new phase in it. First source of bugs is a formal testing phase that includes exploratory testing during development phase, secondly if bugs delivered to customer

Figure 8: Comparison

5.1 Hypothesis

We developed hypothesis (H1: Inclusion of Exploratory and Regression testing makes the Bugs manifestation process more efficient and reliable and HA: Inclusion of Exploratory and Regression testing does not make Bugs manifestation process more efficient and reliable).

On the bases of results, we cannot reject H1. It means that exploratory and regression testing have positive impact on bugs manifestation process and it makes process more efficient and reliable.

5.2 Regression Model

i. $Re = \alpha+\beta1\ (EXPT) + \beta2\ (RET) +\varepsilon$

ii. $Ef = \alpha+\beta1\ (EXPT) + \beta2\ (RET) +\varepsilon$

(Where Re is Reliability, Ef is Efficiency, EXPT is Exploratory Testing and RET is Regression Testing).

We applied regression on model 1 to check the impact of exploratory testing and regression testing on reliability. We have 10 test cases so we encode it with 1-10 and if test case is being then it is encoded with 1 otherwise 0. Results are following

Table 1: Regression Statistics

Regression Statistics	Values
Multiple R	0.552978412
R Square	0.305785124
Adjusted R Square	0.107438017
Standard Error	2.860387768
Observations	10

Table 2: Anova Table

ANOVA					
	df	SS	MS	F	Significance F
Regression	2	25.2272	12.6136	1.5416	0.2787
Residual	7	57.2727	8.1818		
Total	9	82.5			

The result of F-test is 1.542 and it is highly significance as p value is .278 which means that reliability is dependent on exploratory regression testing.

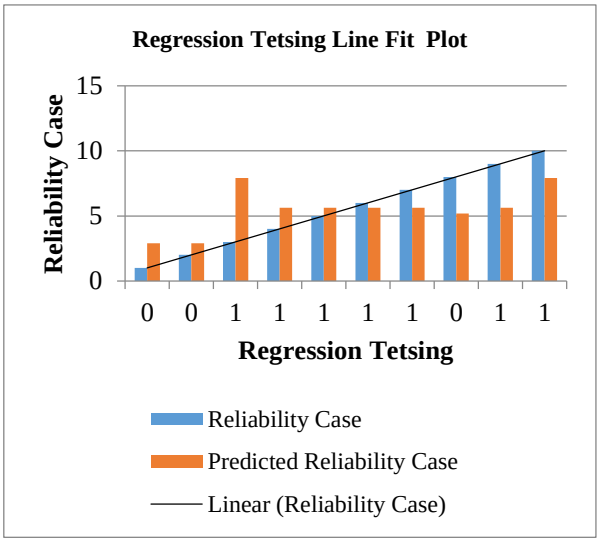

Figure 9: Reliability Dependency On Regression Testing

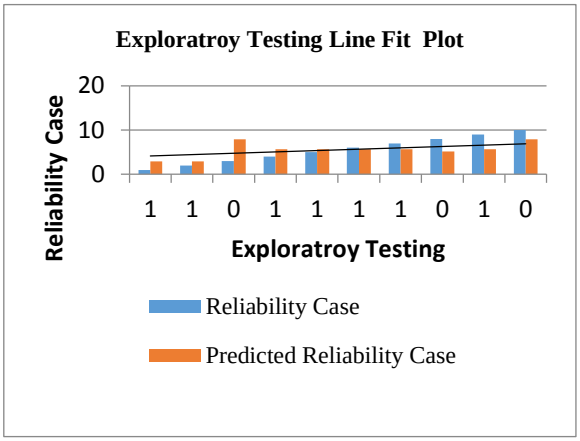

Figure 10: Reliability Dependency On Exploratory Testing

Similarly, we applied regression on model 2 to check the impact of exploratory and regression testing on Efficiency. We have 10 test cases so we encode it with 1-10 and if test case is being then it is encoded with 1 otherwise 0. Results are following: -

TABLE 3: REGRESSION STATISTICS

Regression Statistics	Values
Multiple R	0.552978412
R Square	0.305785124
Adjusted R Square	0.107438017
Standard Error	2.860387768
Observations	10

TABLE4: ANOVA

ANOVA

	Df	SS	MS	F	Significance F
Regression	2	25.2272	12.6136	1.3236	0.2323
Residual	7	57.2727	8.1818		
Total	9	82.5			

The result of F-test is 1.3236 and it is highly significance as p value is .2323 which means that reliability is dependent on exploratory regression testing.

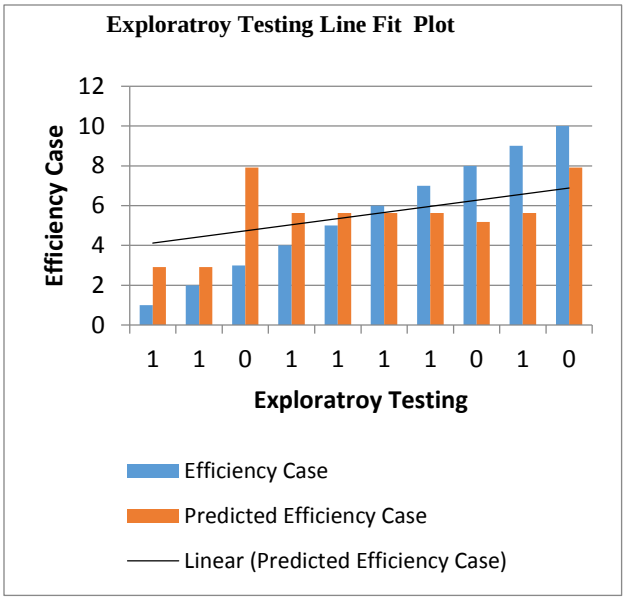

Figure 11: Efficiency Dependency On Exploratory Testing

6. CONCLUSION AND FUTURE WORK

While developing the software projects, software developers ignore the software bugs manifestation process. These bugs do not show up easily at development time. In this research work, the researcher tried to analyze bugs manifestation process to find flaws. Two main processes areas (development and maintenance) were selected after analysis of bugs manifestation process.

After deeply observation of bug's manifestation process we have introduce two new phase in existing model. First source of bugs, i.e. formal testing phase that include exploratory testing during development phase, secondly, if bugs delivered to customer than regression testing will be applied at maintenance phase. These two phases will play a significant role within the reliability of projects and safe it towards failure.

In future work the author will try to apply this model in Enterprise Resource Planning (ERP) system to ensure its usability and reliability in real scenario.

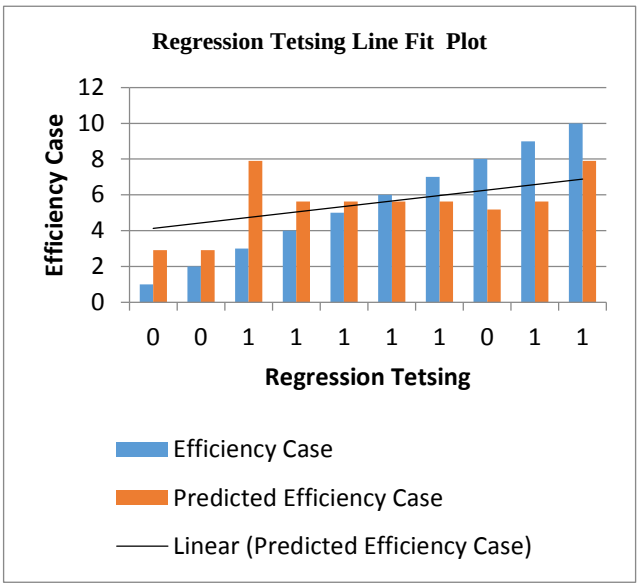

Figure 12: Efficiency Dependency On Regression Testing

REFERENCES

[1] Bovenzi, A., D. Cotroneo, R. Pietrantuono and S. Russo. On the aging effects due to concurrency bugs: a case study on MySQL. 2012 IEEE 23rd International Symposium on Software Reliability Engineering, *IEEE*. (2012)

[2] Chandra, S. and P. M. Chen. Whither generic recovery from application faults? A fault study using open-source software. Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on, *IEEE*. (2000)

[3] Cotroneo, D., R. Natella and R. Pietrantuono. "Predicting aging-related bugs using software complexity metrics." *Performance Evaluation* **70**(3): 163-178(2013).

[4] Grottke, M., A. P. Nikora and K. S. Trivedi. An empirical investigation of fault types in space mission system software. 2010 IEEE/IFIP International conference on dependable systems & networks (DSN), *IEEE*. (2010)

[5] Hall, T., D. Bowes, S. Counsell, L. Moonen and A. Yamashita. "Software fault characteristics: A synthesis of the literature." (2015).

[6] Herzig, K., S. Just and A. Zeller. It's not a bug, it's a feature: how misclassification impacts bug prediction. Proceedings of the 2013 International Conference on Software Engineering, *IEEE Press*. (2013)

[7] Lee, I. and R. K. Iyer. "Software dependability in the Tandem GUARDIAN system." *IEEE Transactions on Software Engineering* **21**(5): 455-467(1995).

[8] Lu, S., S. Park, E. Seo and Y. Zhou. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. ACM Sigplan Notices, *ACM*. (2008)

[9] Sullivan, M. and R. Chillarege. Software defects and their impact on system availability: A study of field failures in operating systems. FTCS. (1991)