

Process of finding defects in software testing

Arti Rana Dept. of Computer Science & Engineering Uttaranchal University Dehradun, India Email: artirana11190@gmail.com Arvind Singh Rawat Dept. of Computer Science & Engineering Uttaranchal University Dehradun, India Email: rawat.s.arvind@gmail.com Anchit Bijalwan Dept. of Computer Science & Engineering Uttaranchal University Dehradun, India Email: anchit.bijalwan@gmail.com

Abstract—Software testing is the most significant stage of the Software Development Life Cycle. Projects underneath testing goes through different stages such as test analysis, test planning, test case, test case review process, test execution process, requirement traceability matrix (RTM), defect tracking (bug logging and tracking), test execution report and closure.

In terms of software, defects means whenever expected results not meet actual results. Generally defect is known as a bug. It talks about the complete life cycle of a bug right from the stage it was found, fixed, re-test, and close.

This paper basically deals with entire process of bug life cycle and how to avoid the bug. To avoid the bug, Test Engineer should prepare the bug report template which consists of various steps.

Index Terms-defects; software testing; bugs; request for enhancement.

I. INTRODUCTION

Software testing is an essential procedure in almost each and every new developing project. It is also known as debugging the software. Debugging is the method of discover and dropping the fault in a computer program. Debugger is a debugging tool which helps to recognizing coding mistakes at various software progress stages.

Software testing is the major activity of estimating and accomplishing product with an observation to find out faults. It is the procedure where the system constraints and system modules are implemented and estimated manually or by using automation tools to figure out whether the system is fulfilling the specific requisites or not [1].

In our research we have acknowledged the implications of number of rejected defects in software products and then accessed the facts over number of incorrect reports that cascade in recognized classes of reasons for bug rejection. Our research provides a statistical analysis over the relationship among number of reported bugs and rejected bugs.

The rest of paper bifurcated as problem statement in section II.

II. PROBLEM STATEMENT

The objective of testing is to discover the problems and provide the better solution to avoid such types of problems.

Software testing typically represents 40% of a software development budget. To start any testing process first of all test engineer should have to prepare test document and test plan.

Before starting the test execution process as a test engineer, we should prepare some necessary documents so that we do not forget the scenarios/flow/values that need to be tested.

We should prepare document so that if old test engineer leaves and the new person comes without wasting much time, he can start testing by referring to test document. Also if he prepared the documents we can maintain test consistency whenever we go for test execution process.

On the other hand, Test plan is a dynamic document that derives entire testing activities. It is prepared in the beginning as soon as the requirement is gathered. Following are the attributes of test plan.

A. Objective: It shows that, what is the aim of writing the test plan. This test plan is written to test the functionality of product.

B. Scope: It consists of two sub part:

(i) inscope i.e, features which needs to be tested.

(ii) out of scope i.e, features which need not to be tested.

C. Test Methodology: It defines the types of testing which needs to be performed by test engineer for start particular release. Testing types are shown as below:

(i) Start with smoke testing (test basic and critical features of an application.)

(ii) Functional testing

(iii) Integration Testing

(iv) System Testing

(v) Regression Testing

(vi) Compatibility Testing

D. Approach, Test Environment, Templates, Roles and Responsibilities, Effort estimation, Entry and exit criteria, Schedule, Automation, Defect tracking, Assumption, Deliverables, Risks, Mitigation/Backup/Contingency plan. Bug life cycle is a complete life cycle of a bug. Whenever a bug is fixed, we retest the bug and change the status accordingly. It might be closed or re-open. We get bugs because of

- Wrong code i.e., logic of the program is wrong or functionality does not work according to the requirement.
- **Missing code** i.e., developer has forgotten to develop that particular feature and it is not available in the application.
- Extra coding i.e., developer has created a feature which is not available in the requirement but exists in application.

Following are the entire process of bug life cycle:



Figure 1.1

STEPS:-

- As soon as we get a bug status of bug is "New/Open".
- The bug is reported to the concerned person by changing the status as "Assigned".
- Once developer gets the bug, first he is going to go through the bug i.e., check if it is valid or not then if it is valid bug start re-producing the bug i.e., perform the actions on the application according to steps mentioned in the bug report and then change the code.
- Once the code changes are done, the developer changes the status to "**fixed**".
- The test Engineer starts re-testing the bug and if it is fixed properly then changes the status to "closed". Else if bug is still exists then status is change to "reopen" and assign it back once again to the developer. This process continues until the bug is "fixed" or "closed".

III. CLASSIFICATION OF DEFECTS

A defect in software testing is an error in programming or logic that causes a program to failure or to produce wrong/unexpected outcome.

Following are the different types of defects/bug:

A. Invalid or rejected bug

Whenever scenario is wrong and developer does not accept it as a bug then it is called invalid/rejected bug.

This bug generally occurred due to the following cases:

Case 1:- if Test Engineer misunderstood the requirement and may log the bug then developer will change the status to **invalid**. Then again Test Engineer go through it and if he also feel that it's invalid and closes the bug by keeping the status as **close**.

Case 2:- if developer misunderstood the requirement and changes the status to **invalid** after go through. The Test Engineer will go through the bug and he also may reproduce it and if the bug exists really he changes the status from **invalid** to **re-open**.

B. Duplicate bug

When same bug is found by different Test Engineer then it is known as **duplicate bug.** This bug is occurred because of following reasons:

- Common features (modules) which is access by both the Test Engineer.
- Dependent features.

Solution for avoid the duplicate bug:

i. Search in bug repository, if bug already exists, do not log (report) a bug. If bug does not exist, log (report) a bug and store in bug repository.

ii. Send it to developer and carbon copy to all test engineers.

C. Not-Reproducible bug

Developer accepts the bug but not able to find the same bug after following the navigation steps mentioned in the bug report.

Reasons for not-reproducible bug:

Platform mismatch

i. Server mismatch

Test Engineer tests the application in one server and developer may reproduce the bug in another server, to avoid this mention server name in bug report.

ii. Environment mismatch

Test Engineer tests the application in different operating system and browser and developer may reproduce the bug in different operating system and browser, to avoid this mention platform name in bug report.

• Data mismatch

Scenario may be correct, but for testing an application, Test Engineer use different data and developer may use different data for reproducing the bug, to avoid this mention test data in bug report.

• Build mismatch

Test Engineer got a bug in one build and developer reproduce the same bug in different/another build due to time constraint it is known as build mismatch.

D. Can't fix bug

Developer accept the bug, also able to reproduce it but not able to perform code changes due to some reasons. These reasons are as follows that's why developer can't fix the bug:

- 1. Core of code (bug is in the core of code)
- 2. No technology support

If there will be major changes in bug then developer can't say that can't fix that bug. **Can't fix bug** should be minor bug but all minor bug cannot be **can't fix bug**.

E. In-consistent bug

In first time Test Engineer found a bug but after that he is not able to find the same bug in next time, so to avoid the inconsistency Test Engineer should take the screenshot and follow the following steps:

- 1. As soon as Test Engineer got the bug first to take the final screenshot of that bug.
- 2. Re-confirm the bug whether it is consistent or not.
- 3. If bug is consistent then search the bug in bug repository. If bug report is not found then prepare the bug report and send it to the developer.

F. Deferred/Postpone bug

Even though there is a bug, it is postpone to the future releases due to time constraint. Due to time constraint developer going to make it to deferred i.e. he will fix it in next release.

In the initial builds, bugs can't be deferred but in the later stages due to the time limit he can deferred the bug. But the test engineer will check that bug can be really deferred or not. Deferred bug can be minor bug but the entire minor bug cannot be deferred. We cannot close the bug until it is fixed in next release.

G. Requests for Enhancement

These are the suggestion given by the test engineer towards the enhancement of the application.

IV.RESULT ANALYSIS

To avoid various types of bug, test engineer should uses relational database for its repository and prepare a test case template as well as bug report template which are shown as follows:

Test Case Template		
HEADER:		
Test Case		
Name/ID:		
Test Case type	Functional/Integration/System	
Requirement #:		
Module:		
Status:		
Severity:	Critical/Major/Minor	
Release:		
Version:		
*Pre-Condition:		
*Test Data:		
Brief Description:		
PODY		
Step#·		
Description:		
Inputs:		
Expected result:		
Actual result:		
Status:		
Comments:		
FOOTER		
Author:		
Date:		
Reviewed by:		
Approved by:		

Table 1.1

BUG REPORT TEMPLATE	
1. Bug #/ID :	
2. Test Case Name:	
3. Module:	
4. Requirement # :	
5. Date:	
6. Reporter :	
7. Assigned to :	
8. Status :	
9. Severity :	
10. Priority :	
11. Server :	
12. Platform :	
13. Test Data :	
14. Build # :	
15. Attachment	
screenshot :	
16. Brief Description :	
17. Steps to reproduce :	
18. Observation :	
19. Expected result :	

In table 1.1, severity means what kind of bug and its impact on application whereas priority means the order in which the bug has to be fixed by the developer team.

Pre-Condition is the minimum data configuration that needs to be created by Test Engineer before starting the test process.

In table 1.2, build defines a piece of software which consists of set of features, bug fixes and needs to be tested for stability or check whether build is stable or not.

V. CONCLUSION

With the help of Testing we can reduce the bug from the software but it cannot prove that there is no remaining bug or software is bug free.

The main reasons for bug rejections are improper bug reports and inadequate information of test engineer over the developed project. To avoid such kind of bug we should prepare a proper bug report. To start the testing process firstly test engineer should have a clear knowledge of requirement. After that he can clearly identify the bug. He should prepare the test document and test plan as per the requirement. Based on these test plan further test engineers can prepare the test case where all the positive as well as negative scenarios are included. And prepare test case, test scenario, bug report template in such a way that any other person can easily reproduce the same scenario or identify the bug without wasting time over reviews and rework. Development team as well as testing team should be a part of requirement gathering team for enhanced understanding of software requirements. It will help the project team to

have a proper proposal of scope and restrictions to avoid rework.

References

- Itti Hooda, et al. "Software Test Process, Testing Types and Techniques", International Journal of Computer Applications (0975 – 8887) February 2015, Volume 111 – No 13.
- [2] Maneela Tuteja, Gaurav Dubey, et al. "A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, July 2012, Volume-2, Issue-3.
- [3] Mohd. Ehmer Khan, et al. "Different forms of software testing techniques for finding errors", IJCSI International Journal of Computer Science, May 2010, Vol. 7, Issue 3, No. 1.
- [4] Aranda, J., & Venolia, G., et al. "The Secret Life of Bugs: Going Past the Error and Omissions in Software Repositories", 31st IEEE International Conference on software Engineering (ICSE), 2009, pp298-308.
- [5] Qin, F., Tucek, J., & Zhou, Y., et al. "Treating Bugs As Allergies: A Safe Method for Surviving Software Failures", Proceedings of the 10th conference on Hot Topics in Operating Systems(HOTOS), 2005, Vol. 10, pp.19-19.
- [6] Breu, S., Premraj, R., Sillito, J., & Zimmermann, T., et al. "Information needs in bug reports: improving cooperation between developers and users", Proceedings of the 2010 ACM conference on Computer supported cooperative work(CSCW), 2010, pp301-310.
- [7] Akhilesh, Babu, Kolluri.K, Tameezuddin.Kalpana, Guddikadula.,et al. "Effective Bug Tracking Systems.Theories and Implementation", IOSR Journal of Computer Engineering ISSN:2278-0661, Oct 2012, Volume 4,Issue 6, pp 31-36.
- [8] Giovanni Denaro, Bernhard Scholz, Zhi Quan Zhou, et al. "Automated Software Testing and Analysis: Techniques, Practices and Tools", 2014 47th Hawaii International Conference on System Sciences, p. 260, 2007, doi:10.1109/HICSS.2007.96.
- [9] Vishawjyoti and P. Gandhi, et al. "A survey on prospects of automated software test case generation methods," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 3867-3871.