International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 8, October - 2012

Implementation Of Effective Defect Tracking System In Software Egineering

Kankanala Raja[1],K.Praveen[2]

[1]*Department of CSE, DRK Institute of Science & Technology, Ranga Reddy, Andhra Pradesh, India*

[2]*Associate Professor*
Department of IT, DRK Institute of Science & Technology, Ranga Reddy, Andhra Pradesh, India

Abstract

Defect Tracking Systems in testing is the process which report the defects and also provides the complete information regarding defects. In the testing phase the tester will identify the defects. Whenever the tester encounter number of defects he adds the defect id and information in the database.The tester reports to both project manager and developer. The defect details in the database table are accessible to both project manager and developer.
The project manager assigns projects to the developers. The developer develops the projects as per customer requirements. The project manager itself assigns the developed applications to the Testers for testing. The tester tests the application and identify the defects in the application. When the tester encounter number of defects, he generates a unique id number for each individual defect. The defect information along with its id are mailed to the project manager and developer. This is defect Report. These are stored in the database. This is useful for further reference.
Defect information includes the defect id, defect name, defect priority, project name, defect location, defect type.This whole process continues until all the defects are got fixed in the application.The defect report is mailed to the project manager and the developer as soon as the defect is identified. This makes that no error will go unfixed because of poor communication. It makes ensure that anyone who needs to know about a defect can learn of it soon after it is reported.
Defect Tracking System plays an vital role in the testing phase. But it supports assigning projects for the developer, tester by the project manager. The Defect Tracking System maintains the different users separately i.e., it provides separate environments for project manager, developer and tester.

1.Introduction

Tracking of defects plays an important role in the testing environment of a software project.
According to [0] Testing is a crucial part of the software life cycle, and recent trends in software engineering evidence the importance of this activity all along the development process. Testing activities have to start already at the requirements specification stage, with ahead planning of test strategies and procedures, and propagate down, with derivation and refinement of test cases, all along the various development steps since the code-level stage, at which the test cases are eventually executed, and even after deployment, with logging and analysis of operational usage data and customer's reported failures. Testing is a challenging activity that involves several highdemanding tasks: at the forefront is the task of deriving an adequate suite of test cases, according to a feasible and costeffective test selection technique. However, test selection is just a starting point, and many other critical tasks face test practitioners with technical and conceptual difficulties (which are certainly under-represented in the literature): the ability to launch the selected tests (in a controlled host environment, or worse in the tight target environment of an embedded system); deciding whether the test outcome is acceptable or not (which is referred to as the test oracle problem); if not, evaluating the impact of the failure and finding its direct cause (the fault), and the indirect one (via Root Cause Analysis); judging whether testing is sufficient and can be stopped, which in turn would require having at hand measures of the effectiveness of the tests: one by one, each of these tasks presents tough challenges to testers, for which their skill and expertise always remains of topmost importance.
Defect tracking is a system that is applied for any system software so that system performs well. In a

survey [14] conducted to software engineers of companies like Mozilla, Eclipse, and Apache found that the information items presented in defect reports have played very important role in fixing defects. Insufficient or improper reports caused delay in fixing defects and thus causing crossing deadlines. The information items that can be found
in the defect reports include screenshots, test cases, expected behavior observed behavior, stack traces and steps to reproduce etc. Generally this is the minimum preferred information needed by engineers to fix defects easily. However, the prior research in this area [1] and [2] revealed that the defect reporters omitted these
essential information fields in their defect reports making them poorly designed reports. Such reports are of little use to developers for the purpose of fixing defects.If your defect report is effective changes are higher that it will get fixed. So fixing of the defect are issue is dependent upon how effectively you report it. These systems are used on a wide scale and are treated as essential repositories that help in finding status of defects and eradicating them instantly. Any one can write a defect report, but not everyone can write an effective defect report. A tester should be able to distinguish between an average defect report and a good defect report. Insufficient or improper reports can cause delay in fixing of defects and thus resulting in extending of the given deadlines. A defect report should clearly specify the characteristics' like having clearly specified defect number, reproducible and be specific. Generally, developers expect descriptions beyond the information found in the present defect reports. Depending upon the defect priority and the severity the testing of the defect is done. Priority is generally set as p1 to p5. P1 as "fixing the defect with high priority" and p5 as "fix when time permits". Severity says the impact of the defect.

A. *DEFINITION*

A Software Defect / Defect is a condition in a software product which does not meet a software requirement (as stated in the requirement specifications) or end-user expectations (which may not be specified but are reasonable). In other words, a defect is an error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results.

- A program that contains a large number of defects is said to be *defectgy*.
- Reports detailing defects in software are known as *defect reports*.
- Applications for tracking defects are known as *defect tracking tools*.
- The process of finding the cause of defects is known as *dedefectging*.
- The process of intentionally injecting defects in a software program, to estimate test coverage by monitoring the detection of those defects, is known as *bedefectging*.

Software Testing proves that defects exist but NOT that defects do not exist.

B. *CLASSIFICATION*

Software Defects/ Defects are normally classified as per:

- Severity / Impact
- Probability / Visibility
- Priority / Urgency
- Related Dimension of Quality
- Related Module / Component
- Phase Detected
- Phase Injected

Related Module /Component

Related Module / Component indicates the module or component of the software where the defect was detected. This provides information on which module / component is defectgy or risky.

- Module/Component A
- Module/Component B
- Module/Component C
- …

Phase Detected

Phase Detected indicates the phase in the software development lifecycle where the defect was identified.

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

Phase Injected

Phase Injected indicates the phase in the software development lifecycle where the defect was introduced. Phase Injected is always earlier in the software development lifecycle than the Phase Detected. Phase Injected can be known only after a proper root-cause analysis of the defect.

- Requirements Development
- High Level Design
- Detailed Design
- Coding
- Build/Deployment

Note that the categorizations above are just guidelines and it is up to the project/organization to decide on what kind of categorization to use. In most cases, the categorization depends on the defect tracking tool that is being used. It is essential that project members agree beforehand on the categorization (and the meaning of each categorization) so as to avoid arguments, conflicts, and unhealthy bickering later.

I. RELATED WORK

Software development project report undergo many challenges like working with different stakeholders in a project, teams would be facing difficulty in transparency of defects due to lack of process for defect logging, standard defect tracking, no clear communication, improper communication process. An error, fault, failure, mistake, flaw can be called as "defect" with respect to a software system. SDLC undergoes many phases. Even though the development team focuses well during all the phases while developing a particular product there may be some mistakes in any phase. In today's market conditions, we have so many freeware tools available for test or defect management tools. Some projects also afford to have cost assigned to defect management tools and buy required amount of licenses. Before, people use to use a simple way for defect tracking by sending mail to a related technical department. This is difficult to keep track the defect as the emails are scattered. So, then testers started using manual testing (manual defect tracking) via email.

When quality for a product is not met, customer gets dissatisfied. According to [7] a defect in software system occurs for a is not accident defect that occurs due to specific reason.

A. Defect Tracking Life Cycle

This defect Tracking Life cycle follows the concept of Bugzilla Defect Life Cycle[16] . The design of the system had been done according to the user requirements and brief analysis of a new system. Each defect in the system will have an status, priority, and severity. The status indicates the current progress with respect to an issue.
Defect tracking is the process of finding defects in a product, (by inspection, testing, or recording feedback from customers), and making new versions of the product that fix the defects.

Defect tracking is important in software engineering as complex software systems typically have tens or hundreds of thousands of defects: managing, evaluating and prioritizing these defects is a difficult task. Defect tracking systems are computer database systems that store defects and help people to manage them.
Using Defect tracking tool the following process is followed:

a.Loggingintothetool
b.DefectLifeCycle
c.Creatingadefect
d.Changingstatusofdefects
e.Generatingmetricsandreports

B. Defect Life Cycle (Defect Life cycle) is the journey of a defect from its identification to its closure. The Life Cycle varies from organization to organization and is governed by the software testing process the organization or project follows and/or the Defect tracking tool being used.

Nevertheless, the life cycle in general resembles the following:

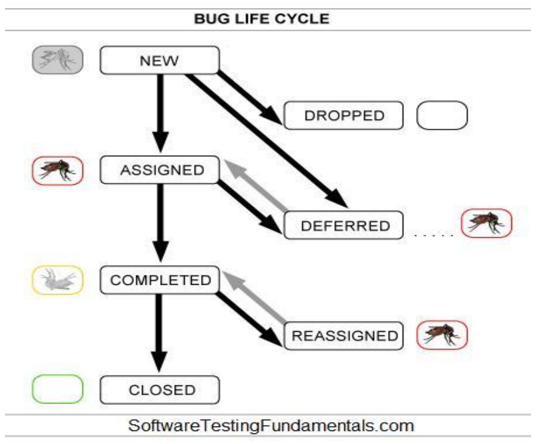

Fig:1 Defect Tracking Life Cycle

Status	Alternative Status
NEW	
ASSIGNED	OPEN
DEFERRED	
DROPPED	REJECTED
COMPLETED	FIXED, RESOLVED, TEST
REASSIGNED	REOPENED
CLOSED	VERIFIED

Defect Status Explanation

- *NEW*: Tester finds a defect and posts it with the status NEW. This defect is yet to be studied/approved. The fate of a NEW defect is one of ASSIGNED, DROPPED and DEFERRED.
- *ASSIGNED / OPEN*: Test / Development / Project lead studies the NEW defect and if it is found to be valid it is assigned to a member of the Development Team. The assigned Developer's responsibility is now to fix the defect and have it COMPLETED. Sometimes, ASSIGNED and OPEN can be different statuses. In that case, a defect can be open yet unassigned.
- *DEFERRED*: If a valid NEW or ASSIGNED defect is decided to be fixed in upcoming releases instead of the current release it is DEFERRED. This defect is ASSIGNED when the time comes.
- *DROPPED / REJECTED*: Test / Development/ Project lead studies the NEW defect and if it is found to be invalid, it is DROPPED / REJECTED. Note that the specific reason for this action needs to be given.
- *COMPLETED / FIXED / RESOLVED / TEST*: Developer 'fixes' the defect that is ASSIGNED to him or her. Now, the 'fixed' defect needs to be verified by the Test Team and the Development Team 'assigns' the defect back to the Test Team. A COMPLETED defect is either CLOSED, if fine, or REASSIGNED, if still not fine.
- If a Developer cannot fix a defect, some organizations may offer the following statuses:
    - *Won't Fix / Can't Fix*: The Developer will not or cannot fix the defect due to some reason.
    - *Can't Reproduce*: The Developer is unable to reproduce the defect.
    - *Need More Information*: The Developer needs more information on the defect from the Tester.
- *REASSIGNED / REOPENED*: If the Tester finds that the 'fixed' defect is in fact not fixed or only partially fixed, it is reassigned to the Developer who 'fixed' it. A REASSIGNED defect needs to be COMPLETED again.
- *CLOSED / VERIFIED*: If the Tester / Test Lead finds that the defect is indeed fixed and is no more of any concern, it is CLOSED / VERIFIED. This is the happy ending.

C. DEFECT SEVERITY CLASSIFICATION

The actual terminologies, and their meaning, can vary depending on people, projects, organizations, or defect tracking tools, but the following is a normally accepted classification.

- *Critical:* The defect affects critical functionality or critical data. It does not have a workaround. Example: Unsuccessful installation, complete failure of a feature.
- *Major:* The defect affects major functionality or major data. It has a workaround but is not obvious and is difficult. Example: A feature is not functional from one module but the task is doable if 10 complicated indirect steps are followed in another module/s.
- *Minor:* The defect affects minor functionality or non-critical data. It has an easy workaround. Example: A minor feature that is not functional in one module but the same task is easily doable from another module.
- Trivial: The defect does not affect functionality or data. It does not even need a workaround. It does not impact productivity or efficiency. It is merely an inconvenience. Example: Petty layout discrepancies, spelling/grammatical errors.

Severity is also denoted as:

- S1 = Critical
- S2 = Major
- S3 = Minor
- S4 = Trivial

Defect Severity is one of the most common causes of feuds between Testers and Developers. A typical situation is where a Tester classifies the Severity of Defect as Critical or Major but the Developer refuses to accept that: He/she believes that the defect is of Minor or Trivial severity.

Though we have provided you some guidelines in this article on how to interpret each level of severity, this is still a very subjective matter and chances are high that one will not agree with the definition of the other. You can however lessen the chances of differing opinions in your project by discussing (and documenting, if necessary) what each level of severity means and by agreeing to at least some standards (substantiating with examples, if necessary.)

D. *Defect Priority* indicates the importance or urgency of fixing a defect. Though priority may be initially set by the Software Tester, it is usually finalized by the Project/Product Manager.

Priority can be categorized into the following levels:

- *Urgent:* Must be fixed in the next build.
- *High:* Must be fixed in any of the upcoming builds but should be included in the release.
- *Medium:* May be fixed after the release / in the next release.
- *Low:* May or may not be fixed at all.

E. ***DEFECT REPORT TEMPLATE***

In most companies, a defect reporting tool is used and the elements of a report can vary. However, in general, a defect report can consist of the following elements.

ID	Unique identifier given to the defect. (Usually Automated)
Project	Project name.
Product	Product name.
Release Version	Release version of the product. (e.g. 1.2.3)
Module	Specific module of the product where the defect was detected.
Detected Build Version	Build version of the product where the defect was detected (e.g. 1.2.3.5)
Description	Detailed description of the defect. Describe as much as possible but without repeating anything or using complex words. Keep it simple but comprehensive.
Steps to Replicate	Step by step description of the way to reproduce the defect. Number the steps.
Actual Result	The actual result you received when you followed the steps.
Expected Results	The expected results.
Attachments	Attach any additional information like screenshots and logs.
Remarks	Any additional comments on the defect.
Defect Severity	Severity of the Defect.
Defect Priority	Priority of the Defect.
Reported By	The name of the person who reported the defect.
Assigned To	The name of the person that is assigned to analyze/fix the defect.
Status	The status of the defect.
Fixed Build Version	Build version of the product where the defect was fixed (e.g. 1.2.3.9)

REPORTING DEFECTS EFFECTIVELY

It is essential that you report defects effectively so that time and effort is not unnecessarily wasted in trying to understand and reproduce the defect. Here are some guidelines:

- ***Be specific:***
    - Specify the exact action: Do not say something like 'Select ButtonB'. Do you mean 'Click ButtonB' or 'Press ALT+B' or 'Focus on ButtonB and click ENTER'? Of course, if the defect can be arrived at by using all the three ways, it's okay to use a generic term as 'Select' but bear in mind that you might just get the fix for the 'Click ButtonB' scenario. [Note: This might be a highly unlikely example but it is hoped that the message is clear.]
    - In case of multiple paths, mention the exact path you followed: Do not say something like "If you do 'A and X' or 'B and Y' or 'C and Z', you get D." Understanding all the paths at once will be difficult. Instead, say "Do 'A and X' and you get D." You can, of course, mention elsewhere in the report that "D can also be got if you do 'B and Y' or 'C and Z'."
    - Do not use vague pronouns: Do not say something like "In ApplicationA, open X, Y, and Z, and then close it." What does the 'it' stand for? 'Z' or, 'Y', or 'X' or 'ApplicationA'?"
- ***Be detailed:***
    - Provide more information (not less). In other words, do not be lazy. Developers may or may not use all the information you provide but they sure do not want to beg you for any information you have missed.
- ***Be objective:***
    - Do not make subjective statements like "This is a lousy application" or "You fixed it real bad."
    - Stick to the facts and avoid the emotions.
- ***Reproduce the defect:***
    - Do not be impatient and file a defect report as soon as you uncover a defect. Replicate it at least once more to be sure. (If you cannot replicate it again, try recalling the exact test condition and keep trying. However, if you cannot replicate it again after many trials, finally submit the report for further investigation, stating that you are unable to reproduce the defect anymore and providing any evidence of the defect if you had gathered.)
- ***Review the report:***
    - Do not hit 'Submit' as soon as you write the report. Review it at least once. Remove any typos.

F. *Defect Detection Efficiency (DDE)* is the number of defects detected during a phase/stage that are injected during that same phase divided by the total number of defects injected during that phase.

DDE = (Number of Defects Injected AND Detected in a Phase / Total Number of Defects Injected in that Phase) x 100 %

$$\text{DDE} = \frac{\text{Number of Defects Injected AND Detected in a Phase}}{\text{Total Number of Defects Injected in that Phase}} \times 100\ \%$$

Defect Density is the number of confirmed defects detected in software/component during a defined period of development/operation divided by the size of the software/component.

$$\text{Defect Density} = \frac{\text{Number of Defects}}{\text{Size}}$$

USES

- For comparing the relative number of defects in various software components so that high-risk components can be identified and resources focused towards them.
- For comparing software/products so that quality of each software/product can be quantified and resources focused towards those with low quality.

II. Initial Experiment

Effective Defect Tracking System was developed by using Spring-MVC frame work.Spring MVC is used to develop the applications quickly. The functionalities of Effective Defect Tracking System is categorized in to the following modules such as Administration, Project Management, BugManagement, Activity and Reports.

In Administration module Project Manager or Administrator adds the employees such as Developers or programmers,Testers or Reporters and Guests or Clients.After providing the employee profile an individual automatically gets the mail from the system so that employee can interact with the system directly.

In project Management module the complete project information like number of modules and number of builds.Where each build refers to a screen of the concerned the project.Project manager or Administrator assigns the Modules of the project to the developer or programmer,and after completion of the Build,administrator assigns particular to the tester in the testing department.

In Bug Management module tester registers the defects and information of test cases ,Steps to Reproduce, relvent StackTrace and Screenshots.In this Module tester assigns the priority and status of the Defect.Below Figure 2 shows the Bug Entry process,where reporter enter the detailed information.

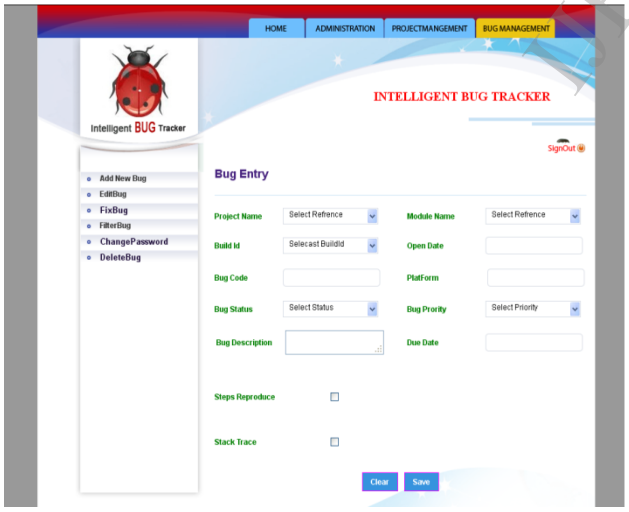

Fig 2– Main UI of Defect Tracking Application

III. Evaluation

The proposed directions for Implementation of Effective Defect Tracking System in Software Egineering are tested using the defect tracking application. The application is capable of tracking defects with sufficient information that leads to fixing defects quickly and efficiently. This results in saving lot of time and expenditure. The team which is developing project will be having an advantage as budget by following the defect effictient process. The development time and cost can be reduced by using defect detection efficiency ,defect density and defect age.

The defect detecting process is very simple by following the concepts like Project based modules and module based builds which refered as Component or BuildID .Each defect is refered to a concerned BuildId and that is directly associate to the Developer.Below figure shows the tracking defect in BugManagement module.

Conclusion

Tracking of defects in a project plays an important role in an overall project life cycle because it ensures the quality of the project.If defect araised in a project, then it must be resolved by using report given by the testing department. The developer develops the projects as per customer requirements. The project manager itself assigns the developed applications to the Testers for testing. The tester tests the application and identify the defects in the application. When the tester encounter number of defects, he generates a unique id number for each individual defect. The defect information along with its id are mailed to the project manager and developer. This is defect Report. These are stored in the database. This is useful for further reference.In future this proposed system can be enhanced by the upcoming technologies like cloud computing etc.

IV. Acknowledgement

Special thanks to my Guide Mr. Praveen K ,Associate Professor , Department of CSE, DRK Institute of Science & Technology,JNTUH.

References

[0] Antonia Bertolino, Eda Marchetti. A Brief Essay on Software Testing.

[1] N. Bettenburg, S. Just, A. Schr¨oter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good defect report? In FSE'08:Proceedings of the 16th International Symposium on Foundationsof Software Engineering, pages 308–318, November 2008.

[2] S. Breu, J. Sillito, R. Premraj, and T. Zimmermann.Frequently asked questions in defect reports.Technical report, University of Calgary, March 2009.

[3] S. Artzi, S. Kim, and M. D. Ernst.Recrash: Making software failures reproducible by preserving object states. In ECOOP'08: Proceedings of the 22nd European Object-Oriented Programming Conference, pages 542–565, 2008.

[4] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim. Duplicate defect reports considered harmful ... really? In ICSM'08: Proceedings of the 24th IEEE International Conference on Software Maintenance, pages 337–345, 2008.

[5] S. Breu, J. Sillito, R. Premraj, and T. Zimmermann.Frequently asked questions in defect reports.Technical report, University of Calgary, March 2009.

[6] Black, R. 1999. Managing the Testing Process: The tools you need. Retrieved January 20, 2010 from http://library.books24x7.com

[7] Limaye. 2009. Software Testing. Retrieved February 06, 2010 from http://books.google.com.my

[8] Robbins, J. 2000. Dedefectging Applications: Getting started dedefectging. Retrieved January 30, 2010 from http://library.books24x7.com.

[9] ZatulAmilahShaffiei, MudianaMokhsin, SaidatulRahahHamidi (2010). Change and Defect Tracking System: AnjungPenchalaSdn. Bhd. International Journal of Computer Applications (0975 – 8887) Volume 10– No.3

[10] Singh, L., Drucker, L. & Khan, N. 2004. Advanced Verification Techniques: A SystemC Based Approach for Successful Tapeout. Retrieved February 05, 2010 from http://books.google.com.my

[11] Smart, J.F. 2007. Javaworld.com: What issue tracking system is best for you? Retrieved February 07, 2010 from http://www.javaworld.com/javaworld/jw-03-2007/jw-03- defects.html?page=1

[12] Barnson, M.P. 2001. The Defectzilla Guide. Retrieved February 10, 2010 from http://db.glugbom.org/Documentation/Defectzilla-Guide/

[13] Craig, R. D. &Jaskiel, S. P. 2002. Systematic software testing.Retrieved February 03, 2010 from http://books.google.com.my.

[14] Akhilesh Babu. K, Tameezuddin .K, and Kalpana Gudikandula. Effective Defect Tracking Systems: Theories and Implementation. In IOSR Journal of computer Engineering, *ISSN: 2278-0661 Volume 4, Issue 6 (Sep-Oct. 2012), PP 31-36*

[15] Goldin, L. &Rochell, L. 2002. Software Development Defect Tracking: "Tool Isn't User Friendly" or "User Isn't Process Friendly". Retrieved February 11, 2010 from http://www.springerlink.com.newdc.oum.edu.my/content/4he0d2ehj6m0y5kv/?p=111918f3b5434b9c979278dae45d93e0&pi=2

[16] Defectzilla Defect Life Cycle. 2007. Defect Life Cycle. Retrieved June 15, 2010 from http://www.softwaretestinghelp.com/defect-life-cycle/

[17] Change and Defect Tracking System: Anjung Penchala Sdn. Bhd. International Journal of Computer Applications (0975 – 8887) Volume 10– No.3, November 2010.

[18] http://softwaretestingfundamentals.com/