Proposed Framework for Quality Assurance System with Duplicate Bug Detection

Aniruddha Kshirsagar
ME Student, Department of Computer Engineering
Flora Institute of Technology,
Pune,India
ani.kshirsagar@rediffmail.com

Pankaj Chandre
Asstt. Prof. Department of Computer Engineering
Flora Institute of Technology,
Pune,India
pankajchandre30@gmail.com

***Abstract*:-** When project are having so cost. Many times the problem of bug will get occur. So, it becomes very important to have proper quality assurance system(QAS).Poorly designed quality assurance systems may exchange wrong information between developers. The purpose of this paper is to make understandings of different quality assurance systems and explain them, to find out problems present in them and give proper direction for improvement so as attract customers, raise customers satisfaction, to reduce downtime .This Paper proposes a framework to detect duplicate bug. detection, QAS, bugs**.**

*****

I. INTRODUCTION

Sachin, a real person: Netbeans 6.0crashed.
Sourav, a bug-tracking system: What did you do?
Sachin: I clicked on File ! New Project and OK.
Sourav: Did you choose a Java project?
Sachin: No.
. . . (a few questions later)
Sourav: Thanks Sachin. The bug is Homepage.java
and we will fix it soon.
This example shows need of Bug/Quality Assurance
System.

The use of bug tracking systems[3] as a tool to organize maintenance activities. The systems serve as a central repository for monitoring the progress of bug reports, requesting additional information from reporters and discussing potential solutions for fixing bug. Developers use the information provided in bug reports to identify the cause of the defect, and narrow down plausible files that need fixing.

What is Bug,Virus?

Bug[2] - A software bug is an error, mistake, failure, or fault
in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result). Most bugs arise from mistakes and errors made by people in either a program's source code or its design, and a few are caused by compilers producing incorrect code.

Virus[2] - A computer virus is a computer program that can copy itself and infect a computer without the permission or knowledge of the owner and attaches itself to a program or file enabling it to spread from one computer to another, leaving infections as it travels. Like a human virus, computer virus can range in severity: some may cause only mildly annoying effects while others can damage your hardware, software or files. Almost all viruses are attached to an executable file, which means the virus may exist on your computer but it actually cannot infect your computer unless you run or open the malicious program. It is important to note that a virus cannot be spread without a human action, (such as running an infected program) to keep it going. Because a virus is spread by human action people will unknowingly continue the spread of a computer virus by sharing infecting files or Sending emails with viruses as attachments in the email. Sometimes when you try to send a mail with an attachment of .exe file to another then it fails, this is due to a virus detection in email server Items such as stack traces, steps to reproduce, observed and expected behavior, test cases, and screenshots ranked high on the list of preferred information by developers.

There is a mismatch between what developers consider most helpful and what users provide[5].A quality assurance system is an application that lets to keep track of bugs for software project in database. Duplicate bug report entries in QAS impact negatively on software maintenance , productivity[4].

II. LITERATURE SURVEY

Nicolas Serrano and Ismael Ciordia [4] has compared two bug tracking tool i.e. are Bugzilla and ITracker. The objective of their research was to provide a comparative study of these two bug tracking tool based on the criteria platform independence, database independence, how customizable is it, are the number of users limited and life of cost. G Abaee and D.S Guru[5] gave the best practice to test documentation and effort estimations have been investigated as well as Bug Tracking Tools. They compare the four different existing Bug Tracking Tools with each other along with their features and drawbacks. Then they proposed new one, the Debugger

Thomas Zimmermann, et al. [6] addressed the concerns of bug tracking systems by proposing four broad directions for enhancements .They discussed that it is important that information provided in bug reports is relevant and complete in order to help resolve bugs quickly.. Poorly designed bug tracking systems are partly to blame for exchange of information being stretched over time enhancements. As a proof-of-concept, they also demonstrate a prototype interactive bug tracking system that gathers relevant information from the user and identifies files that need to be fixed to resolve the bug.

Fischer et al. [10] discussed that Version control and bug tracking systems contain large amounts of historical information that can give deep insight into the evolution of a software project. Unfortunately, these systems provide only insufficient support for a detailed analysis of software evolution aspects. They addressed this problem and introduced an approach for populating a release history database that combines version data with bug tracking data and adds missing data not covered by version control systems such as merge points.

S. Just et al. [11] concluded that Developers typically rely on the information submitted by end-users to resolve bugs. They conducted a survey on information needs and commonly faced problems with bug reporting among several hundred developers and users of the ECLIPSE, APACHE and MOZILLA projects.

M.P. Francisco et al. [12] have developed a tool to extract and to store information from Debian's BTS (Bug Tracking System) in a relational database. In this paper ,they showed that there is a strong dependence between three variables which can be used to analyze the activity of a project through its bugs: communications between users and developers, bug notifications and people involved. They explained that bugs are an essential part of software projects because they lead its evolution. Without bug notifications developers cannot know if their software is accomplishing its tasks properly.

A. Hora et al. [13] discussed that to harness the complexity of big legacy software; software engineering tools need more and more information on these systems. This information may come from analysis of study of execution traces, the source code, computing of metrics, etc. One source of information received less attention than source code: the bugs on the system. Little is known about the evolutionary behavior, lifetime, distribution, and stability of bugs. In this paper, they proposed to consider bugs as first class entities and a useful source of information that can answer such topics.

Stephen Blair in his paper [14] provided tips and guidelines for evaluating features, and explains how these features fit into a defect tracking process. He discussed that evaluating a bug tracking system requires that you understand how specific features, such as configurable workflow and customizable fields, relate to your requirements and your current bug tracking process. He explained before you start evaluating bug tracking systems; make sure you identify your requirements for the system.

The life cycle of a bug can be as follows:
1. New: If there is any bug, its status is considered as New.
2. Assigned: Here, bug is assigned to particular developer.
3. Resolved: When bug get solved ,it is treated as resolved.
4. Verified: Test results provided by the developer are verified by the tester.
5. Closed: Bug is finally closed after the proper solution is provided.
6. Reopened: If the QA team is not satisfied by the solution provided for the bug, it is reopened.

Table 1 shows facilities provided in each Quality Assurance System.

Table 1: Classification between Quality Assurance System[1]

	Search	Email alerts	Reports	Charts	Time Tracking ,Free
Bugzilla	yes	yes	yes	yes	yes
Mantis	yes	no	no	no	yes
BugTracker.NET	yes	yes	yes	yes	yes
Redmine	yes	yes	yes	yes	yes
Bugzero	yes	yes	yes	no	no

1) **Bugzilla** very popular, actively maintained and free bug tracking system, used and developed together with Mozilla, giving it considerable credibility. Bugzilla is based on Perl and once it is set up, it seems to make its users pretty happy. It's not highly customizable, but in a odd way, that may be one of its features: Bugzilla installations tend to look pretty much the same wherever they are found, which means many developers are already accustomed to its interface and will feel they are in familiar territory. Bugzilla has a system that will send you, another user, or a group that you specify the results of a particular search on a schedule that you specify. Bugzilla has a very advanced reporting systems and you can create different types of charts including line graph, bar graph or pie chart.QAS is also known as Defect Tracking System[7].

2) **Mantis** is a free web-based bug tracking system. It is written in the PHP scripting language and works with

MySQL, MS SQL, and PostgreSQL databases and a webserver.

Mantis can be installed on Windows, Linux, Mac OS and OS/2. Almost any web browser should be able to function as a client. It is released under the terms of the GNU General Public License (GPL). The main complaint is its interface which doesn't meet modern standards. On the other hand, is easy to navigate, even for inexperienced users. There not exists some advanced features such as charts and reports. In short, the whole system is sloppily done, there are plenty of bugs and very little functionality.

3)**BugTracker.NET** is a free, open-source, web-based bug tracker or customer support issue tracker written using ASP.NET, C#, and Microsoft SQL Server Express.
BugTracker.NET is easy to install and learn how to use. When you first install it, it is very simple to setup and you can start using it right away. Later, you can change its configuration to handle your needs. It has a very intuitive interface for generating lists of bugs. It has two very useful features. First of them is a screen capture utility that enables you to capture the screen, add annotations and post it as bug in just a few clicks. The second feature is the fact that it can integrate with your Subversion repository so that you can associate file revision check ins with bugs.

4) **Redmine** is a flexible web-based project management web application. Written using Ruby on Rails framework, it is cross-platform and cross-database. Redmine is open source and released under the terms of the GNU General Public License. Redmine is flexible issue tracking system. You can define your own statuses and issue types. He support multiple projects and subprojects. Each user can have a different role on each project. Interface is very simple, intuitive and easy to navigate. Shortly, this is very good product and our recommendation.

5) **Bugzero** is a web-based bug, defect, issue and incident tracking software. Its single code base supports both Windows and Unix (based on Java™) and supports database systems including Access, MySQL, SQL Server, Oracle, and etc. Bugzero can be customized for software bug tracking, hardware defect tracking, and help desk customer support issue and incident tracking. Bugzero have intuitive interface but he lacks form features. The main drawback is the fact that Bugzero is an commercial product and you can find much better product for free.

III. HOW TO IMPROVE QUALITY ASSURANCE SYSTEM

Having complete information in the initial bug report (or as soon as possible) helps developers to quickly resolve the bug. The focus of our work is on improving bug tracking systems with the goal of increasing the completeness of bug reports. Specifically, improving bug tracking systems is done in four ways.

1)***Tool Centric***[18]-:It means that the QAS can be configured to collect stack trace implicitly and add it to the report that contains bug details. It can improve the information collection capabilities. It can also use steps to make use of capture/replay tools that can be observed by software engineers later.

2)***Information Centric***[18]-:This is another direction from us that helps software developers to have improved focus on the collection of information that has to be kept in bug reports. Such tools verify the information provided in bug reports and provide feedback which helps to improve the quality of information.

3)***Process Centric***[18]-:Process centric feature helps developers to estimate time to be spent on particular bugs and schedule their time accordingly.

4)***User Centric***[18]-:Users centric feature includes both developers and bug reporters. This focuses on educating the reporters so as to enable them to collect proper information and how to collect it as well. The expected information in the bug report makes the developers to grasp it faster and act quickly to fix bugs in real time applications.

IV. QUESTIONS ASKED IN BUG REPORT

When any software engineer presents a bug report, most probably, he is asked many questions. Some of them are what is the name of the product? What is the bug? in which component is the bug?; in which module is the bug?; in which method the bug is?; in which environment the bug arises?; in which platform the application is built?; in which OS the application runs?. The information given by developer who report bug might be incomplete initially. For this reason follow up questions required. They include do you have .NET framework installed? Can you provide a screenshot of the bug? Research revealed that the software developers may answer 2 questions out of three [19]. When a bug report is submitted by a developer, the follow up questions are to be asked immediately besides keeping the submitted bug report in hand. We recommend software development teams to have bug tracking systems that contain "build expert systems". These systems ask all required questions to software engineer so as to make the work automated. The question to be given and answered by developer is not static. The questions do not come sequentially. Moreover answer to a question determines the next possible question. Narrowing down the location of bug and to have accurate bug descriptions are features of expert. The following data is essential in order to build an expert system. Bug location information which is crucial to tacking bugs. Location gives you the line number, method, class and so on. This helps developers to move to that place with ease. Many software development environments (IDEs) allow the bugs to be located just by a click of button or a click. From the bugs list, machine learning models can be built that choose questions and also predict the location of the bug based on the responses that corresponds to the bugs. This paper provides a proof of study that makes use of data that is present in the bug reports. Thus we get collection of information that is essential in implementing a tool that can support automatic evaluation of the information.

V. FRAMEWORK FOR QUALITY ASSURANCE SYSTEM

Fig. 1 shows flowchart of proposed quality assurance system.

This system will also guide the manager to allocate specific type of bug (by means of bug details) to specific developer . This technique is very advantageous when the developer gets similar bug for solving (which kind of he has already solved before) where he has expertise. Otherwise, other developer has to research or to discuss on that bug and then go for solution, which might be time consuming and result could be less than accurate.

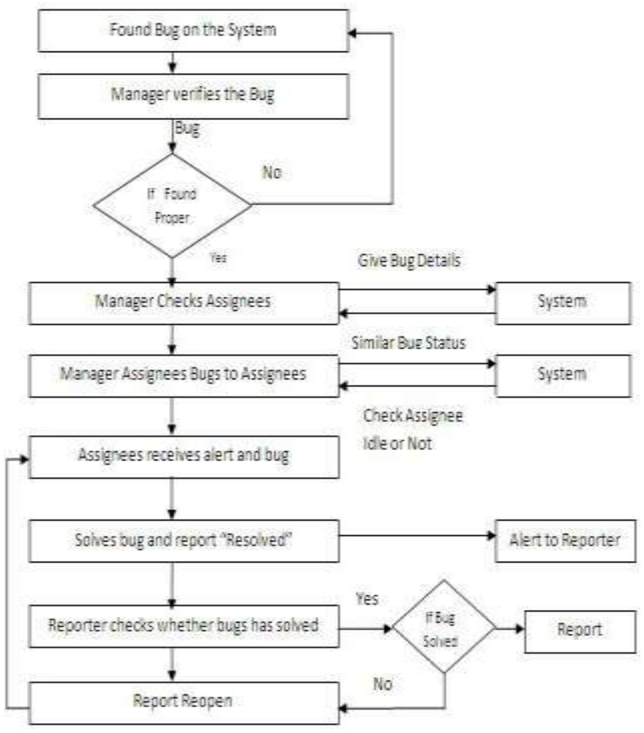

Fig.1: Flowchart of Proposed Quality Assurance System

Steps will be as follows.
1)Project go through SDLC model and then it found bug on proposed system.
2)Manager verifies the bug/Alert comes to manager.
3)If it is a bug, then manager checks for an assignee and give similar bug details to the system.
4)Manager assigns bug to assignee after checking whether assignee is idle or not.
5)Then, Assignee receives alert and bug details also.
6)Assignee solves the bug and report it as “resolved”.
7)This Alert goes to reporter.
8)Reporter checks whether bug has been solved or not.
9)If bug has not been solved then control again goes to step no.5.
10)If bug has been solved, report get displayed.

In proposed system, there will be a facility as “Detecting Bug Duplicate” for Quality Assurance Systems. There may be huge number of duplicate bug reports .In some projects, as many as a quarter of all reports are duplicates. Initially, developers have to manually identify duplicate bug reports, but this process is very time-consuming .Proposed system will automatically classify duplicate bug reports as they arrive to save developer time.

CONCLUSION

Proposed framework for QAS will be helpful to raise customer’s satisfaction, increase productivity and also to minimize maintenance cost.

ACKNOWLEDGMENT

Sincere thanks to the all authors, whose papers in the area of Software Engineering published in various conference, proceedings and journals.

REFERENCES

[1] Trajkov Marko, Smiljkovic Aleksandar "A Survey of Bug Tracking Tools: Presentation, Analysis and Trends"
[2] Rajnish Kumar et al, "Improving Software Quality Assurance Using Bug Tracking System" (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 (3) , 2013, 492-497
[3] Thomas Zimmermann,Rahul Premraj,,Jonathan Sillito,Silvia Breu, “*Improving Bug Tracking Systems*”, ICSE’09, May 16-24, 2009, Vancouver, Canada 978-1-4244-3494- 7/09/$25.00 © 2009 IEEE.
[4] Yguarat˜a Cerqueira Cavalcanti, Eduardo Santana de Almeida et.al "An Initial Study on the Bug Report Duplication Problem" 14th European Conference on Software Maintenance ,2010
[5] Thomas Zimmermann, Rahul Premraj et.al.,"What Makes a Good Bug Report?"Ieee Transactions On Software Engineering, VOL. 36, NO. 5, September/October 2010
[6] V.B. Singh1, Krishna Kumar Chaturvedi, "Bug Tracking and Reliability Assessment System (BTRAS) "International Journal of Software Engineering Vol. 5 No. 4, October, 2011
[7] A.S.Syed Fiaz, N.Devi, S.Aarthi "Bug Tracking and Reporting System" International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-1, March 2013

[8] Dr. P.K.Suri1,Rajni Rana "Defect Analysis and Prevention Techniques for Improving Software Quality" 128X International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 7, July 2013 ISSN: 2277

[9] Nawagata Nilambari, Shivani Gautam,Vintee Chaudhary, "*A Survey on Automated Duplicate Detection in a Bug Repository*" International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 3 Issue 4, April - 2014

[10] M. Pinzer Fischer, H. Gall "Populating a Release History Database from version control and bug tracking systems" Software Maintenance, IEEE, 2003.

[11] S, Just, R. Premraj and T. Zimmermann. "Towards the next generation of bug tracking systems", Visual Languages and Human-Centric Computing, IEEE, 2008.

[12] M.P Francisco, P.B. Perez and G. Robles "Correlation between bug notifications, messages and participants in Debian's bug tracking system" Empirical Software Engineering and Measurement, First International Symposium, 2007.

[13] A. Hora, N. Anquetil, S. Ducasse, M. Bhatti, C. Couto, M.T. Valente and J. Martins, "Bug Maps: A Tool for the Visual Exploration and Analysis of Bugs" Software Maintenance and Reengineering (CSMR), 16th European Conference, 2012.

[14] Stephen Blair "A Guide to Evaluating a Bug Tracking System, White paper, 2004.

[15] Nicolas Serrano, Ismael Ciordia, "Bugzilla, ITracker, and Other Bug Trackers", IEEE software, Vol 22, pp. 11-13, 2005.

[16] G Abaee, D.S. Guru, "Enhancement of Bug Tracking Tools; the Debugger", Software Technology and Engineering (ICSTE), 2010

[17] Nicholas Jalbert, Westley Weimer "Automated Duplicate Detection for Bug Tracking Systems" International Conference on Dependable Systems & Networks: Anchorage, Alaska, IEEE, 2008.

[18] 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.

[19] S. Breu et. Al Frequently asked questions ,Calgary University2009