IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 4, Issue 6 (Sep-Oct. 2012), PP 31-36
www.iosrjournals.org

Effective Bug Tracking Systems: Theories and Implementation

Akhilesh Babu Kolluri[1], K. Tameezuddin[2], Kalpana Gudikandula[3]
[1]*Department of CS, DRK Institute of Science & Technology, Ranga Reddy, Andhra Pradesh, India*
[2,3]*Department of CS, DRK College of Engineering & Technology, Ranga Reddy, Andhra Pradesh, India*

Abstract: *Bug tracking is an essential discipline in the domain of software engineering. It has far reaching effects on the system when effectively used. The information provided in terms of bugs and solutions in the bug reports can help software engineers to act on them quickly and ensure that they are either rectified or eliminated from the system. The bulk of information provided in the bug reports may cause problem to developers in ascertaining poorly designed information. Therefore the bug tracking systems are to be improved and follow certain standards. To overcome the problem, we propose four fundamental directions to enhancing effectiveness of bug tracking systems. To demonstrate the efficiency of the proposed directions, we develop a prototype application that tracks bugs effectively by capturing essential information from users and help resolve bugs quickly.*

I. Introduction

Bug tracking is a system that is indispensable for any system that has to perform well. It is a tool that facilitates fixing of bugs faster and ensures the quality of software being developed or being used. These systems are widely used and they are treated as essential repositories that help in finding status of bugs and quickly resolving them thus the progress of the project can be ascertained. The potentiality and soundness of a good bug tracking system has no parallel in helping the systems to enhance the quality to meet the expectations of clients. Software engineers frequently use bug reports and try to fix them if there is enough information required. In a survey [1] conducted to software engineers of companies like Mozilla, Eclipse, and Apache found that the information items presented in bug reports have played very important role in fixing bugs. Insufficient or improper reports caused delay in fixing bugs and thus causing crossing deadlines. The information items that can be found in the bug 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 bugs easily. However, the prior research in this area [1] and [2] revealed that the bug reporters omitted these essential information fields in their bug reports making them poorly designed reports. Such reports are of little use to developers for the purpose of fixing bugs. When developers need very descriptive information and the bug reports lack such information, it leads to stalling of the project or delay in completion with other cause and effects. The main problem with current bug tracking systems is that they are simply storing some information fields into databases and not descriptive in nature. Developers expect descriptions beyond doubt and they are not found in the present bug reports. The proposed work in this paper addresses this problem by providing solid and usable steps or directions that can practically improve understandability of information and also ensure that essential data has been captured and stored in bug reports. We also show the results of a prototype bug tracking application and its results and efficiency in helping software developers to understand and fix bugs quickly. The four directions proposed by us are tool oriented improvements, information oriented improvements, process oriented improvements and user oriented improvements. They are elaborated in the later sections.

II. Related Work

Software development involves many challenges. One such challenge is the process of bug tracking. Bugs in the process of software development are unavoidable. However, they are to be tracked and fixed with some focus otherwise they are manifested into the final product and that leads to failure of software systems. An error, fault, failure, mistake, flaw can be called as “bug” with respect to a software system. SDLC has many phases. The development team may commit mistakes in any phase. However, when comes to coding it gets surfaced once execution is tested. Most of the bugs can be avoided by having correct analysis and design as per customer requirements. According to [6] bugs in software system cause the system to fail. It does mean that such software system can’t meet the quality and expectations of the client. When software system is able to meet the functional requirements given by the customer, which is said to be a quality product. When quality problem arises, customer gets dissatisfied. According to [7] a bug in software system is not an accident bug that occurs due to specific reason. Tracking bugs has many phases and it has its own life cycle as shown in fig. 1.

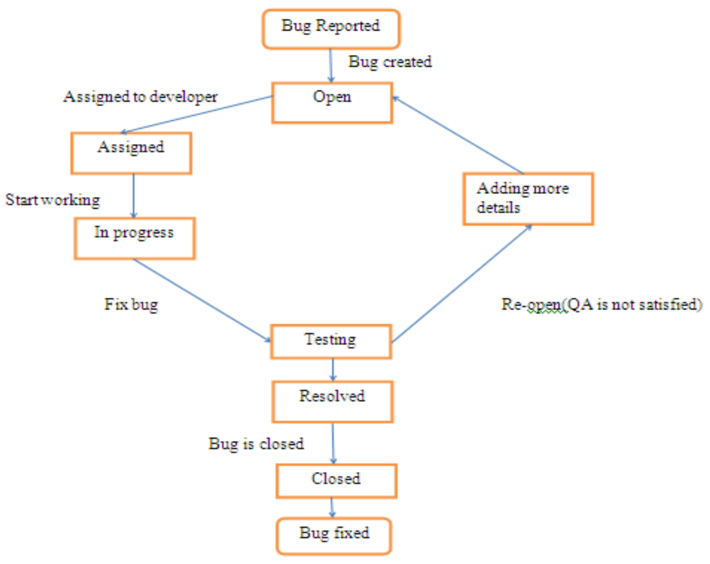


Fig. 1 – Bug Tracking Life Cycle [9]

As can be seen in fig. 1, an bug tracking is cyclic in nature. Right from its initial finding, until it is resolved or fixed permanently it has its lifecycle with certain phases. As bugs cause systems to act incorrectly bug tracking systems reduce the likelihood of having bugs in the system long time. Quickly the software development team can fix bugs and close them with suitable bug tracking system. Bug tracking improves quality of software being developed. Even after delivering software products bugs may be unearthed and still the tacking system is to be used to fix those bugs faster. When software system is being developed bug tracking is important and also challenging task [14]. Traditionally bug tracking is made simple. It does mean that many years down the line, software developers used spreadsheets to store bug details and fix them. However, using software like spreadsheet is not effective considering modern facilities such as emailing; RSS feeds, and change notification through SMS and so on. Nevertheless, it is very useful to use a spreadsheet application like Excel for small scale projects with less complexity and less direction. As spreadsheets lack in security and not sophisticated when it comes to database operations, now a day's people are using relational databases to store bug reports and present them through a bug tracking system front end. Bug tracking system helps QA teams in software development cycles.

Software development companies certainly use a bug tracking system as it is essential. Bug tracking system is really required while developing every software product as many developers do not maintain adequate documentation with respect to customer requirements through the entire life cycle of the product [8]. There are many benefits of using a bug tracking system. As specified in [6], clear communication can be provided by a bug database. When compared to informal emails etc. well written reports with standards explain the bugs and the priorities very well. However, it is not easy to track bugs. It is a tedious task as specified in [6]. Many bug tracking systems are already available. They are either open source or commercial. Best bug tracking system among them can be used to fulfill customer requirements. When bug tracking systems are used many factors are to be considered. While using bug tracking system the factors to be observed are application setup, reporting process, and notification method. Before using a bug tracking system, it has to be set up. The well known bug-management application which is open source is "Bugzilla". In many open source products such as Linux, Eclipse, and Mozilla it is being used. There might be configuration problems with open source products. According to [10] free bug tracking systems like Bugzilla take more time to setup and do not appear user friendly. This is because its setup process is very complicated. Users feel it difficult while installing its server.

As specified in [11] Bugzilla is difficult to install and maintain though it has rich set of features. Its user interface is also not inspiring. Besides, for a novice user it is not an easy task to install and use Bugzilla as observed in [12]. Even though steps are provided to install and use it, it is never easy to get it working for the first time. With respect to setup process it is not an easy task with respect to open source products like Bugzilla. It has to be made simple and easy for novice users as well. With respect to reporting process, bug reporting should be made very simple and efficient bug tracking is expected. The steps in the bug reporting have to be simplified to have smooth communication of bugs among the team of software development. When a web application has so many HTML pages and not tuned to have good navigation, it is difficult to work with such system. When it come to bug tracking on such web applications, problems reoccur as there are navigation problems prevailing in the system and people who are involved in bug tracking gets frustrated with the application. The complexity in such systems can be reduced by working out a smooth navigation and thus developers feel easy to work with bug tracking of such systems. There are some bugs tracking systems that make the reporting process a very tedious task. Instead of reducing burden on people, they really increase it as the reporting process is so complex.

Any bug tracking system can be of two types such as simple and complicated. However, ideally it must be simple and efficient. It should work fast and help people to track and fix bugs so as to develop quality products and satisfy clients. "Bug Genie" is another real world bug tracking system with very good features. However, its bug reporting process is not straightforward and so complicated. Moreover users who are involved in software development will have to add so many things to bug report that makes it much more complex. According to [13] a bug tracking system should help software engineers to effectively track bugs and thus develop a quality product that can satisfy their customers. Another important feature a bug tracking system should have is notifications. Even when software engineers are not logged into system, they are to be notified about certain events and that must be part of overall workflow of the system. The notifications help team members to understand the changes in bug reports and the timelines to make themselves available. Notifications are part of automated bug tracking system. For instance, when a new bug is reported, all stakeholders pertaining to bug tracking are to be notified immediately. Many bug tracking systems like "BugTracker.NET" and "Flyspray" send mails as notifications. However, spam filters sometimes may block them thus making email notifications unreliable. RSS (Really Simple Syndication) feeds as provided by Web 2.0 technology is another way of making notifications. When users get notifications through RSS feeds, spam filters do not block them. According to [14], content subscribers of RSS get feeds automatically and thus the stakeholders of bug tracking process are aware of notifications. When RSS is used effectively and correctly users are quickly notified about changes and they need not have to be with a system always. This is the reason that a bug tracking system should have facility of RSS feeds that maximize the communication efficiency among the members who are involved in bug tracking. This paper contributes to the research of bug tracking systems yet in another way. It has proposed specific and effective directions to enhance the efficiency of bug tracking systems.

III. Effective Bug Tracking Systems

Software engineers often involve in fixing bugs in the system under development. The time they spend on that can be reduced and the quality of software can be increased by using an effective bug tracking system. The initial information on a bug can help the engineers to resolve bugs faster thus saving development time and cost. Over a period of time development team can reduce lot of wastage of time with regard to tracking and fixing bugs with a sophisticated bug tracking system in hand. This fact motivates us to build a conceptual framework that provides required directions that can improve bug tracking systems. Out proposed framework is as given in fig. 2.

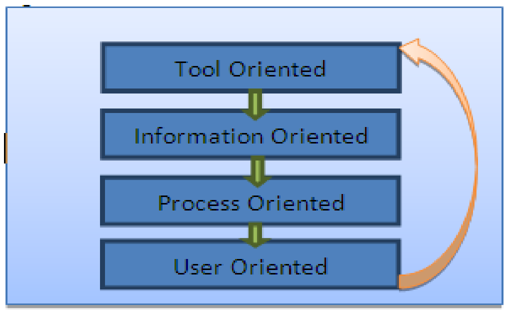


Fig. 2- Proposed Directions to Enhance Bug Tracking Systems

As can be seen in fig. 2, the directions proposed are tool oriented, information oriented, process oriented and user oriented in nature. They are iterative directions that can be cycled back after each iteration.

3.1 Tool oriented

We recommend a bug tracking system to be tool oriented. Tool oriented does mean that the bug tracking systems 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 by doing so. It can also use steps to make use of macro recorders, capture/replay tools that can be observed by software engineers later. From the observations, it is possible to capture test cases and fix the bugs easily [3]. Tool oriented feature can enhance the capability of bug tracking system in terms of information collection that is quite relevant and can be readily used to fix bugs in the system. This will lead the effective tracking and fixing of bugs those results in quality of the software besides being productive. This direction can helps in transition into information oriented feature.

3.2 Information oriented

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. Towards this end some sort of tools like Cuezilla [4] can be embedded into bug tracking systems. Such tools verify the information provided in bug reports and provide real world feedback that helps improve the quality of information. This will help software engineers to get motivated and have more focused work in solving or fixing bugs. Being information oriented with tool support can enhance the possibilities of checking whether the stack traces reported are complete and consistent in the given scenario. This direction when followed by real world bug tracking systems can lead to make it more process oriented.

3.3 Process oriented

This direction is meant for process oriented improvements. The process being followed in bug tracking systems can be improved further using this direction. Is does mean that all administrative activities pertaining to bug tracking and fixing come under process oriented feature. The process oriented bug tracking systems can also focus on the developer who is made responsible to fix bugs besides having a comprehensive bug report [1], [4]. Other advantages of process oriented feature are that developers can have better awareness on bug reports and thus they are aware of possible actions to be taken; it also helps developers to estimate time to be spent on specific bugs and schedule their time accordingly. Process oriented feature can influence user-oriented feature.

3.4 User oriented

Users do mean the developers and also bug reporters. This direction focuses on educating the reporters so as to enable them to collect correct information and how to collect it as well. This training helps both developers and reporters. The expected information in the bug report makes the developers to grasp it faster and act quickly to fix bugs in real time applications. This direction can influence the adaption of new tools in the process thus making it more robust and productive in nature.
The above mentioned directions provide a systematic approach in improving bug tracking systems. To support this we have built a prototype application that makes use of all directions and enhance the capabilities of bug tracking systems in the real world.

IV. Questions 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 [5]. 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.

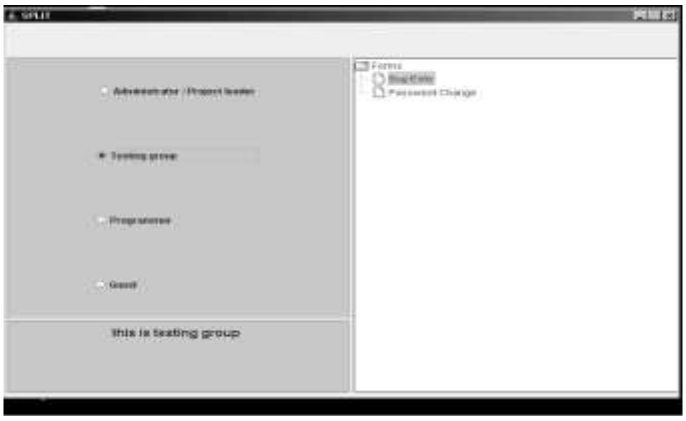
  This paper provides a proof of study that makes use of data that is present in the bug reports. Thus we get collection of information that is essential in implementing a tool that can support automatic evaluation of the information.

4.1 Initial Experiment

We have done an experiment with Eclipse where a set of questions are supposed to predict the accurate location of a bug that can be fixed. We used around 20 most fixed files in the SDK of Eclipse and selected all bug reports of them. From those bug reports, more than 2500 cases are considered and a decision tree is built on the prediction of location of the bug fix of the name of the file. The following questions are used for which answers exist in the bug database of Eclipse.

- What is your name?
- What is the operating system?
- Which version of Eclipse is used?
- What is the bug?
- How severe the bug is?
- Which component is affected by the bug?
- What is the priority to be given to the bug?
- Which platform is affected?

The result of this experiment is a decision tree where the root node shows the defect distribution among all files. The decision tree helps in locating the file which contains the reported bug. We have built an application to demonstrate good features of a bug tracking system that follows the four directions discussed in this paper. The proposed prototype application has the user interface as shown in fig. 3.



this is testing group

Fig. 3 – Main UI of Prototype Application

The application facilitates various stakeholders of the project to record bug details and track them leading to quick and efficient bug fixes. The tool is able to cater various groups of the software development personnel.

V. Valuation

The proposed directions for improving bug tracking systems are tested using the prototype application. The application is capable of tracking bugs with sufficient information that leads to fixing bugs quickly and efficiently. This results in saving lot of time and expenditure. The development team can avoid over budget and crossing deadlines. The development time and cost can be reduced. The tool oriented nature of the system reduces the burden of information collection as it can automatically locate stack traces and add the comprehensive data to bug report. The information oriented nature of the application facilitates integration of tools for obtaining real time feedback. The process oriented nature of the application allows easy administration activities pertaining to bug tracking. The user oriented nature of application educates developers and bug reporters on which kind of information to be provided for easy bug tracking.

VI. Conclusion

Software Development Lifecycle (SDLC) can never be completed without encountering bugs in the coding phase. The bugs thus surfaced are to be fixed faster and go ahead with development. Bug tracking needs necessary information to resolve bugs faster. The present systems for bug tracking are effective. However, they can be further improved by following certain standards. It does mean that bug tracking systems can be improved. This paper has done this. We proposed a framework with four directions that can help improve bug tracking systems. They are tool oriented, information oriented, process oriented and user oriented in nature. By following these directions, ideal bug tacking systems can be built. The directions are iterative in nature and one or more loops of directions while following make the bug tracking systems perfect. The more interactions followed while building a bug tracking system, the more effectively it works. To demonstrate the effectiveness of the proposed system, a prototype bug tracking system is built that asks relevant questions as per the enhanced framework with directions. The answers to the questions make the process of fixing bugs faster. Thus it fosters the bug fixing process with improvement in time taken to fix bugs. The proposed framework can further improved in future by making it more robust and interactive and adapt to all kinds of software systems.

References

[1] N. Bettenburg, S. Just, A. Schr¨oter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug 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 bug 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 bug 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 bug 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. Debugging Applications: Getting started debugging. Retrieved January 30, 2010 from http://library.books24x7.com.

[9] ZatulAmilahShaffiei, MudianaMokhsin, SaidatulRahahHamidi (2010). Change and Bug 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- bugs.html?page=1

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

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

[14] Goldin, L. &Rochell, L. 2002. Software Development Bug Tracking: "Tool Isn't User Friendly" or "User Isn'tProcessFriendly".RetrievedFebruary11,2010fromhttp://www.springerlink.com.newdc.oum.edu.my/content/4he0d2ehj6m0y5kv/?p=111918f3b5434b9c979278dae45d93e0&pi=2

About Authors

	Akhilesh Babu Kolluri is a student of DRK Institute of science and Technology, Ranga Reddy, Andhra Pradesh, India. He has received B.Tech degree in Computer Science and Engineering and M.Tech Degree in Computer Science. His main research interest includes Software Engineering, Enterprise Resource Planning.
	Tameezuddin. K is a student of DRK College of Engineering & Technology, Ranga Reddy, Andhra Pradesh, India. He has received B.Tech Degree in Computer Science and Engineering and M.Tech Degree in Computer Science. His main research interest includes Software Engineering, Networking.
	Kalpana Gudikandula is working as Associate Professor at DRK Institute of Science & Technology, Ranga Reddy, Andhra Pradesh, India. She has received M.Tech Degree in Computer Science. Her Main Interest includes Cloud Computing, Software Engineering.