# A REVIEW ON BUG TRACKING SYSTEM USING NAÏVE BYES IN DATA MINING

**[1]Er. Amandeep, [2]Er. Saurabh Mittal**

*[1]M. Tech. Scholar, CSE Department,*
*Galaxy Global Imperial Technical Campus, Dinarpur Ambala (India)*
*[2]Associate Professor, CSE Department,*
*Galaxy Global Group of Institutions, Dinarpur, Ambala, (India)*

## ABSTRACT

*For many years, bug-tracking mechanism is employed only in some of the large software development houses. Our aim is to distinguish the very fast and the very slow bugs in order to prioritize which bugs to start with and which to exclude at the mean time respectively. We used naïve Bayes classifier to compute our prediction model of the data of four systems taken from three large open source projects Mozilla, Eclipse and Gnome. Conventional procedure of simply relying on shared lists and email to monitor the status of defects is error-prone and tends to cause those bugs judged least significant by developers to be dropped or ignored.*

*Bug(Defect) Tracking System allows individual or groups of developers to keep track of outstanding bugs in product, solution or an application effectively.*

*The Bug Tracking System can dramatically increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance. Some salient features are …*

1. *Product and Component based*
2. *Creating & Changing Bugs at ease*
3. *Query Bug List to any depth*
4. *User Accounts to control the access and maintain security*
5. *Simple Status & Resolutions*
6. *Multi-level Priorities & Severities.*
7. *Targets & Milestones for guiding the programmers.*

## I INTRODUCTION

The purpose of Bug Tracking for improving software reliability is to provide better service to the administrator or useful for applications developed in an organization. In a BTS, some Bug Reports(BR) are labeled by bug reporters as security bug reports (SBRs), whose associated bugs are found to be security problems. SBRs generally deserve higher fix priority than not-security bug reports (NSBRs), the subset of BRs that are believed not to have a security

impact. Correctly labeling SBRs among BRs submitted to a BTS is important in security practice since delay of identifying and fixing the security bugs involved in the SBRs causes serious damage to software-system stakeholders. The likelihood of unlabeled SBRs in a BTS could be high for at least three reasons.

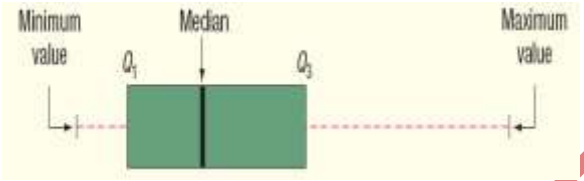


**Fig 1.1**

First, if bug reporters perceive a subtle security bug that they are reporting in a BR as an innocuous not-security bug, then they may label the BR as an SBR. Second, some security bugs described in BRs are associated with recommended mitigations that may be unknown to bug reporters. Third, a bug related to general reliability problems can also be related to security problems and a bug reporter without sufficient security knowledge may report this bug as a NSBR. Bug reporters may often mislabel SBRs as NSBRs due to lack of security domain knowledge as discussed earlier. Then it is desirable for security engineers to inspect NSBRs submitted to a BTS to identify SBRs that are manually-mislabeled as NSBRs in the BTS. However, manually inspecting NSBRs in a BTS to identify SBRs is time consuming and infeasible or not even conducted in practice.

Therefore, there remains a strong need of effective tool support for reducing human efforts in this process of identifying SBRs in a BTS, enabling this important security practice of SBR identification in either industrial or open source settings

### 1.1. Scope

The Bug Tracking System is a window based application that can be accessed throughout the organization. This system can be used for logging bugs against an application/module, assigning bugs to team members and tracking the bugs to resolution. There are features like email notifications, user maintenance, user access control, report generators etc in this system.Bug - A bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from behaving as intended.

### 1.2. Overview

Bug tracking is the process of reporting and tracking the progress of bugs from discovery through to resolution. Other terminology frequently used to describe this process is problem tracking, change management, fault management or trouble tickets.Bug tracking systems are commonly used in the coding and testing phases of the software development.

### 1.3 Authenticate User

The BTS first activates the login form. Here the credentials are authenticated with the existing ones in database. After successful authentication the system activates menus. The activity log also prepared for failures and security.

## 1.4. Products

### 1.4.1. List of Products

The user is provided with the list of existing products. Here the user can view the details of products, modify the existing products and add new products.

### 1.4.2. Product Versions and Users

It is not possible to complete the whole project in a single version. Features required for the product are categorized into several version with dead lines of completion. Here the user can add new versions to a product or can modify the existing details of version.

All employee names and qualifications are stored in the database. Each user is allotted to the product based on their rating, Qualification and designation. For each user Effective date is stored which specifies the total period a user is valid for that product.

## 1.5. Bug Details

### 1.5.1 Bug Details

As the number of bugs for a product can be very large this system is provided with efficient filtering of bugs based on the priority, database, operating system and status. After the user applies filter the list

### 1.5.2. Bug Assignee

This displays the list of users for whom the bug is assigned for resolution since several user are assigned to find a solution for the bug. The user can modify the existing user details.

### 1.5.3 Bug Attachments

While you add a bug you need to provide with the details of bug. So the file attachments can be a document, database file or an image file. Here the user can add a new attachment or can change the details of existing files.

## 1.6 Bug Tracking

### 1.6.1 Track Hierarchy

There might be bugs which can be related to the earlier bugs so our system is provided with a hierarchy. And user can add child nodes in this hierarchy based on the parent child relation ship between the bugs.

### 1.6.2 Track Resolution

This displays a list of all solutions provided by the users allotted to a bug.

## 1.7 Track Resources

As the bugs need to be resolved resources are provided for the bugs based on the rating of the employee.

### 1.7.1 View

### 1.7.2 Product Bug Hierarchy

Here the bugs are displayed in the form of parent child nodes. As it is difficult for the user to look at the vast number of bugs in the database and one cannot easily access the relation between the bugs.

### 1.7.3 Product User Hierarchy

This module is for displaying the users along with their name and designation allotted to the bug. This module simplifies the hierarchy among the employees as well.

## 1.8 Search

Generally Number of bugs for a project increased tremendously so if we want to know about a particular bug it takes much amount of time. With the search screen provided one can filter the bug's base on priority, product, severity, database and type of operating system. He can also list the bugs between particular time based on the start date and end date.

## 1.9 Admin

### 1.9.1 Users

All the users of this system are displayed in this module. One can add new user or can update the details of an existing user. This module saves the details like address, phone and email.

### 1.9.2 Configuration

All the Values like status, priority and others that we are using in this system are configurable.If the user wants to add a new value he need not come to the developer of the product.System is provided with the feature of adding values from the screen.

## 1.10 Log View

In order for the efficient Tracking of the system logs are maintained. The Log View module can be searched based on the user and Records between a start date and end date.

### 1.10.1 Logout

On Log out First the session variable is killed and then redirected to the login page.

### 1.10.2 Prepare Logs

At all the stages, whenever user performs an operation by clicking a button, automatically the Bug Tracking System logs the activity.

We use a four level taxonomy of artifacts derived from activity theory from the work of Engeström and Miettinen to recognize the various artifacts, including organizational processes, that sustain a workflow process. Some examples of organizational processes from our study include bug review meetings (coordinated by "Bug Meisters"), issue meetings (such as "Hit Squad" meetings), and informal discussions during lunch or in the hallway. These artifacts include basic tool artifacts, how-to artifacts, why artifacts and where-to artifacts.

Our study of Apple's Bug Management process revealed that artifacts which facilitate work are often less obvious, including for example, voice mail, email, informal or non- mandated work practices, impromptu meetings. Software bug management is facilitated by more than just the "basic software tools".

### 1.11 Apple's Bug Tracking System

The highly complex and collaborative nature of bug management made for an interesting field study to understand how and what artifacts really get used for software management. Artifacts such as the bug logging and tracking systems are what we view as the "basic software tools".

According to an Apple engineer we interviewed, bug fixing makes up 33% of the effort of a typical software development process at Apple-a huge share of the software development process. Bug management at Apple is viewed as a process of recording, tracking, resolving all bugs throughout the course of development.

### 1.12 Apple Artifact System

Our study of Apple's bug-management indicates that colorful and rich set of colloquialisms used throughout the bug-management process are phrases such as: Hit-squads, Bug-Meisters, Hot-potatoes, Bug-Review Boards, Beta-blockers, and Quality War room. These buzz words describe bug-management artifacts (or tools) that crucially support workflow.

### 1.13 Accurate Bug Tracking System

We will move from the current prototype of the interactive system to a full-scale system that can deal with a variety of information to gather.

One of the most well-known bug tracking systems is Bugzilla. It is an open source project used to track bugs of other open source projects. Typically, the work process is open as well, so one can follow the discourse of the community in their e-mail, chat, or bug tracking conversations.

Bugzilla is a database for bugs. It lets people report bugs and assigns these bugs to the appropriate developers. Developers can use Bugzilla to keep a to-do list and prioritize, schedule and track dependencies.Enter the tasks you're planning to work on as enhancement requests and Bugzilla will help you track them and allow others to see what you plan to work on. If people can see your flight plan, they can avoid duplicating your work and can possibly

help out or offer feedback.It provide summary of the bug tracking process as a collaborative learning process and identify the stakeholders, end-users, designers, implementers, and management. Additionally, sometimes a "bug-czar" or "quality assurance" person facilitates the processing of a bug through its "life cycle." Finally, while anyone could theoretically fix a bug, there is often a small group of individuals responsible for portions of code.

Bug-tracking is a highly socialized process which requires constant communication between developers and bug reporters. However, the inherent social structure of bug tracking systems and its influence on everyday bug-tracking has earlier been poorly studied.Using publicly available information from bug-tracking system database, I model bug reporter reputation to check whether there is any evidence of relation between reporter reputation and attention from developers his bugs get.

### 1.14 Existing System

The existing system consists of entering the details in the Microsoft Excel Sheets for the storing of the data. When a manager needs information of the employee he searches for the specified file in the file system. He opens the file and takes the information.

#### 1.14.1 Limitations in Existing System

- Information retrieval is a very big process.
- No security because the files are visible to the users.
- Report generation will be a big task.

## II LITERATURE SURVEY

This model is reminiscent of Ferree et al.'s "representative liberal" form wherein the media serves the purpose of ensuring the accountability of the representatives via transparency. Yet, different communities interpret the meaning of votes differently. Or, as Brey (1997) argues technical systems are subject to "different interpretations, not only of its functional and social-cultural properties but also of its technical content, that is, the way it works" (Brey 1997).

Asa Dotzler (2002), has stated, "Votes aren't ignored but at the same time they're not the deciding factor in what gets fixed." Those who file bug reports are a tiny fraction of all Mozilla users are probably not representative of the larger community. Furthermore, the voting scheme is simplistic (e.g., users can't vote against a feature).

The comparison to elections and advertising is truly astonishing given that campaign reform and an attempt to end undue influence returning to a "one person, one vote" ideal has been at the forefront of politics for years (Laffoon 2003). Hooimeijer and Weimer [8] observed that bug reports with more comments get fixed sooner. They also noted that bug reports that are easier to read also have shorter life times More recently, Aranda and Venolia have examined communication that takes place between developers outside the bug tracking system [2]. In our earlier work, we discussed shortcomings of existing bug tracking systems [9], which led to the four areas of improvements

presented in this paper. Asking the right questions, is a crucial part of debugging and several techniques such as algorithmic debugging [7] and the WhyLine tool [10] support developers in doing so.

Perry et al., for example, studied individual developers' perceived and actual time allocation for various activities throughout their day-to-day work. Their studies found that over half of each developer's time was spent interacting with co-workers. Her work identified not only the technical axis of knowledge collaboration undertaken by software developers, but also its social axis.Their study found that developers at Microsoft "reported spending nearly half their time fixing bugs." Although much analysis has been done on software development in general, little if any of this attention has been directed toward the in-depth study of how issue or bug tracking systems fit into the role of facilitating communication and collaboration, especially within the context of small, collocated teams that are still the norm in many organizations.

The locations of the bugs are determined by, in which stage the bug appeared or discovered; and in which place in the system it appeared. Fry and Weimer (2010) defined fault localization as: "The task of determining if a program or code fragment contains a defect, and if so, locating exactly where that defect resides" (9). As we attempt to enhance the quality control in the software, we have to recognize different phases of the software development such as: Analysis, Design, Testing and Maintenance. At the research context, we will concentrate only on the phases: (Maintenance and Testing).

The 'Bug Type' varies from one system to another because the different tools which were used to create such systems, have their own limitations and shortcomings. However, we have to put a dynamic framework for defining the bug type according to the various tools used to build the systems; with respect to the major general bug type.

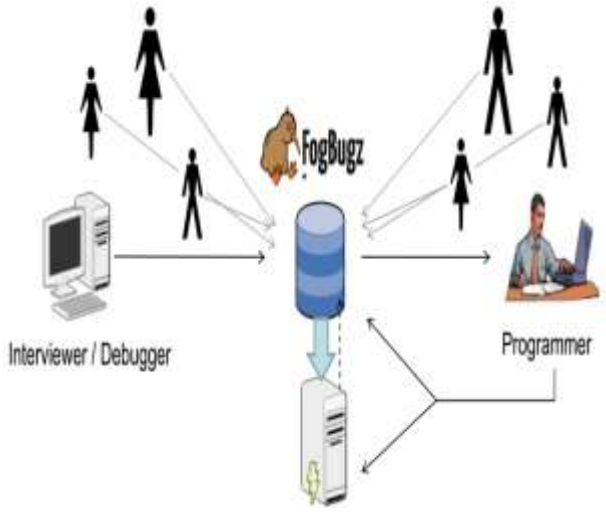


**Fig 2.1: Work Flow Diagram**

According to the work and efforts made by Mays, Jones, Holloway and Studinski, at IBM 1990 for defects prevention. They analyzed the faults that appeared using casual analysis. In addition, detecting the way of prevents

defects from appearing again. They showed the role and importance of the action team whose responsibility was to detect, store and check the appeared faults in the database. Also the important role of triage team mentioned by Black (1999), who assured that the triage team can review, evaluate the defects and assigning them to the development team (3). Cordeiro et al. propose a system which integrates Eclipse IDE (Integrated Development Environment) and Stackoverflow. com (Q&A web resource) [6]. Cordeiro et al. experiments with a context-based recommendation approach and demonstrates that the proposed solution outperforms a simple keyword-based method [6].

B. Code Editors and Web Search Integration Brandt et al. believe that integration of code editors and search tools has significant value and describe a tool called as Blueprint which is implemented (integrating web search) as an extension to a development environment [4]. Their system Blueprint (accessing online example code from within the development environment) is integrated with Adobe Flex Builder, uses Adobe Community Help search API and indexes Flex-specific content. They present experimental results which demonstrates that a integration of development environment with web-search is helpful for the programmers [4].

Goldman present a system called as Codetrail that connects the Eclipse IDE and Firefox Web Browser motivated by the need to integrate web resources into the programming workflow and into the project itself [7]. Codetrail automates some of the manual interactions of developers with web resources and consists of features such as automatic detection and connection of documentation, creation of links, or bookmarks, from code in the development environment to relevant web sites [7].

Hartmann et al. describe a IDE extension called as HyperSource that tracks developers activities in the IDE (such as source code edits) and associates web-pages with the code-edits [8]. HyperSource associates browsing histories with source code edits and displays sets of Web pages that were read while code was edited [8]. Sawadsky et al. describe a tool called as Fishtail which is implemented as a Eclipse IDE plugin to support programmers in discovering code examples and documentation on the web relevant to their current software engineering task [10].

C. Code Search and Recommendation Using Stackoverflow Zagalsky et al. describe a code search and a recommendation tool called as Example Overflow which mines information present in Stack Overflow (Q&A website for programmers) [12]. Their work is motivated by the need to minimize context switch between development environment and code-search tools.

In context to existing work, the study presented in this paper makes the following novel contributions:

a. The work described in this paper is the first focused study on integration of issue tracking systems with community driven question and answering websites such as Stackoverflow. While there has been work in the area of code-editors & development environment integration (Section II-A) with Stack overflow as well as code-editor integration with web-search and external websites (Section II-B), the integration of Stack overflow with issue tracking systems is a unique research direction.

b. We present experimental results indicating presence of several links to Stack overflow question & answers facilitating the process of bug resolution. We present our perspective on the correlation between Stack overflow references and mean time to repair a bug.

c. We present a solution based on analyzing textual features (textual similarity between bug report title and Stackoverflow questions) and contextual features (such as question tags representing the topic) to recommend a Stackoverflow question in response to a bug report. We believe that a recommendation engine that automatically suggests relevant Stackoverflow knowledge base to developers can save time during bug resolution..

d. We believe that there is a dearth of academic studies surveying the needs, problems encountered,human-factors and suggestions on the problem area discussed in this paper.

Open source means the source code is being shared with everybody under the General Public License (GPL) policy. Anyone can contribute to the code voluntarily. There is no restriction towards the submission of code. The moderator will see and verify the reputation of the code submitter through his code submission pattern and its status can also be changed as moderator. While in closed source community, the source code is the property of the organization and the people who are outside the project may not be able to see/browse the code. Outsiders can submit only bugs through the feedback or e-mail to the sales/support person specified by the organization. In this article, we will consider the open source project development scenario except one or two closed source products. Some of bug tracking tools are from open source communities and some of them from closed source communities or commercial organizations. There are organizations which can also provide support for the open source solutions.

### 2.1 The Naive Bayes Probabilistic Model

Abstractly, the probability model for a classifier is a conditional model P(C|f …..f ) 1, n over a dependent class variable with a small number of outcomes or classes, conditional on several feature variables f1 through Fn. The application of the naive Bayes classifier to Spam filtering was initially proposed bySahami, Dumais, Heckerman, and Horvitz (1998), who considered the problem in a decision theoretic framework given the confidence in the classification of a message. A particularly appealing characteristic of a Bayesian framework is its suitability for integrating evidence from different sources. In this sense, Sahami et al. investigated the use not only of the message words, but also application-specific knowledge, in the form of rules regarding the appearance of certain phrases (e.g., ‘‘Free money”), referred to as phrasal features, and non-textual features, obtained through the analysis of the message’s header (e.g., the time when the message was sent). Experiments with two private corpora, using binary features selected according to the information gain, indicated the advantage of including the non-textual features, along with the feasibility of the naive Bayes filter, due to its low false positive rates.

### 2.2 Preprocessing Bug Data

The removed parts of the original web log data that are not relevant in our mining process. After that we get the web log data as 192.168.0.161 7/3/2008/12:00:05 http:// www.google.com/

1. 192.168.0.161 is the IP address (client) that can be used to mind personal usage and the result can be applied in Personalized Systems, Recommender Systems and Pre-fetched System.
2. Web Site Developers and Web Site Analyzer to know the most usage and the least usage date time. http://www.google.com/ is the domain name. Our analyzer is to know which IP frequently uses which sites for which purposes.
3. Web Usage based Mining Mining Specific Site – to support for Web Site Maintainers, Web Site Developers and Web Site Analysts (Statisticians).
4. Mining Specific Client – to know the clients' frequent usage behavior, site and purposes. This mining result can be especially applied in Personalized Systems. □ Mining Specific

### 2.3 Determination

Our system can analyze the most usage site for a particular purpose. From the Web Site Maintainers' and Web Site Developers' point of view, they can prepare to attract more and more clients for their sites.

Fischer et al. mapped program features to the source code where they were implemented, and then tracked the code changes against problem reports involving those features. They then visualize the established relationships to search for feature overlap and dependencies. Such visualization of the evolution of features across time can then be used to find locations in the code where there may be erosion in the software architecture of the system, indicating future problem spots for software maintenance.

Bowman and Holt have analyzed which developers worked on each file in a software system to determine its ownership architecture. The ownership architecture complements other types of architectural documentation. It identifies experts for system components, and can be used to infer the project's internal organization into sub-teams. The ownership architecture can also show non-functional dependencies.

## III CONCLUSION AND FUTURE SCOPE

Current bug tracking systems do not effectively elicit all of the information needed by developers. Without this information developers cannot resolve bugs in a timely fashion and so we believe that improvement to the way bug tracking systems collect information are needed.

While implementing a range of improvements from discussed areas may be ideal, bug tracking systems may instead prefer to specialize, thus providing a rich set of choices. Identify information needs in a large sample of bug reports through manual inspection. This will help to compile a catalog of questions that can be used for the expert system.

Using this catalog, collect answers and defect locations for another large sample of bug reports. This dataset will be used to automatically learn a prediction model.Evaluate the predictions and conduct usability studies. One of the challenges for this project will be to find a representative catalog of questions that is able to predict defects.

## IV FUTURE SCOPE

In addition, scalability will become an issue once more questions with many unique values are used. we will move from the current prototype of the interactive system to a full-scale system that can deal with a variety of information to gather, as commonly observed in the real world.

## REFERENCES

[1] Bugzilla http://www.bugzilla.org/

[2] I. H. Witten, E. Frank, M. A. Hall, "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann; 3rd edition, 2011.

[3] E. Giger, M. Pinzger, and H. Gall. “Predicting the fix time of bugs”,2nd Inter national Workshop on Recommendation Systems for Software Engineering (RSSE’10), May 4, 2010, Cape Town, South Africa ,ACM.

[4] R. L. Scheaffer, M. Mulekar, J. T. McClave, Probability and Statistics for Engineers, Duxbury Press, 5th edition, 2010.

[5] P. Hooimeijer and W. Weimer, “Modeling bug report quality”, In Proc. of the Int’l Conf. on Autom. Softw. Eng., pages 34–43, New York, NY, USA, 2007. ACM.

[6] L. D. Panjer, “Predicting eclipse bug lifetimes”. In Proc. Of the Int’l Workshop on Mining Softw. Repositories, page 29,Washington, DC, USA, 2007. IEEE Computer Society.

[7] P. Bhattacharya, L. Neamatiu, “bug fix time predicition models:can we do better.” http://www.cs.ucr.edu/~pamelab/bugfixtime.pdf

[8] G. Jeong, S. Kim, and T. Zimmermann, "Improving Bug Triage with Bug Tossing Graphs" Proc of the ESEC-FSE, 2009.

[9] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai, "Have Things Changed Now?: An Empirical Study of Bug Characteristics in Modern Open Source Software." Proc of the 1st Workshop on Architectural and System Support For Improving Software Dependability, pp. 25-33, 2006.

[10] A. Podgurski, D. Leon, P. Francis, W. Masri, M. Minch, J. Sun, and B. Wang, "Automated Support for Classifying Software Failure Reports" Proc of the ICSE, pp. 465-475, 2003.

[11] V. Raghavan and M. Wong, "A Critical Analysis of Vector Space Model for Information Retrieval," Journal of the American Society for Information Science, vol. 37, no. 5, pp. 279-287, 1986.

[12] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of Duplicate Defect Report Using Natural Language Processing" Proc of the ICSE, pp. 499-510, 2007.

[13] J. Viega and G. McGraw, Building Secure Software How to Avoid Security Problems the Right Way, Boston, Addison-Wesley, 2002.