Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 3, March 2016, pg.403 – 408

A PROJECTED SOFTWARE REPLICA FOR CREEPY-CRAWLY TRACKING SYSTEM

R.Ramya, Ms. A.Divya

Department of Computer Science, IFET, Villupuram

ABSTRACT: The bugs are followed in a record like Excel, Word, and so on. It has part of issues like time wastage, legitimate specialist not allotted. So as to decrease this correspondence ought to build up between the clients. With the goal that it is difficult to demonstrate to it to the all authorities of that association. So we have diminished the workload and spare time to carry out this employment. This undertaking gives an interface to impart between the workers through neighborhood. The administrator who needs to issue a round can flow it all workers, specific division, specific candidate,... He can see/erase the current round to recreate it. Since it is an online venture it is anything but difficult to utilize and sending. While creating programming in programming concern, the engineer might get mistakes. The engineer needs to invest some energy in new mistakes to discover the arrangements. After some period, the same mistake might happen for another worker. He additionally ought to invest some energy to fathom it. To uproot these sorts of mistakes our application has an interface to include the answer for a blunder. It includes a blunder with mistake sort, mistake number, blunder depiction and arrangement.

Keywords– Bug tracking, User identity, Bug solution

1. INTRODUCTION

A bug tracking system or defect tracking system is a software application that keeps track of reported software bugs in software development projects. It may be regarded as a type of issue tracking system.

Many bug tracking systems, such as those used by most open source software projects, allow end-users to enter bug reports directly. Other systems are used only internally in a company or

organization doing software development. Typically bug tracking systems are integrated with other software project management applications.

A bug tracking system is usually a necessary component of a good software development infrastructure, and consistent use of a bug or issue tracking system is considered one of the "hallmarks of a good software team".

A major component of a bug tracking system is a database that records facts about known bugs. Facts may include the time a bug was reported, its severity, the erroneous program behavior, and details on how to reproduce the bug; as well as the identity of the person who reported it and any programmers who may be working on fixing it.

Typical bug tracking systems support the concept of the life cycle for a bug which is tracked through the status assigned to the bug. A bug tracking system should allow administrators to configure permissions based on status, move the bug to another status, or delete the bug. The system should also allow administrators to configure the bug statuses and to what extent a bug in a particular status can be moved. Some systems will e-mail interested parties, such as the submitter and assigned programmers, when new records are added or the status changes.

The main benefit of a bug-tracking system is to provide a clear centralized overview of development requests (including both bugs and improvements, the boundary is often fuzzy), and their state. The prioritized list of pending items (often called backlog) provides valuable input when defining the product road map, or maybe just "the next release".

In a corporate environment, a bug-tracking system may be used to generate reports on the productivity of programmers at fixing bugs. However, this may sometimes yield inaccurate results because different bugs may have different levels of severity and complexity. The severity of a bug may not be directly related to the complexity of fixing the bug. There may be different opinions among the managers and architects.

A local bug tracker (LBT) is usually a computer program used by a team of application support professionals (often a help desk) to keep track of issues communicated to software developers. Using an LBT allows support professionals to track bugs in their "own language" and not the "language of the developers." In addition, an LBT allows a team of support professionals to track specific information about users who have called to complain — this information may not always be needed in the actual development queue. Thus, there are two tracking systems when an LBT is in place.

2. LITERATURE SURVEY

Bug Tracking and Reliability Assessment System (BTRAS), V.B. Singh, Krishna Kumar Chaturvedi, 2011

Tracking of a reported bug for fixing is a fascinating area of research in software engineering. Many open source, free and commercial bug tracking tools have been developed and are currently under development. The industry needs criteria to select the best tool among the available set of tools that will help in fixing and tracking the progress of bug fixes. In this paper, we use BugZilla, Jira, Trac, Mantis, BugTracker.Net, Gnats and Fossil for comparative study. We present a comprehensive classification criteria to review the available tools and propose a new tool named Bug Tracking and Reliability Assessment System (BTRAS) for the bug tracking/reporting and reliability assessment. BTRAS helps in reporting the bug, assigning the bug to the developer for fixing, monitoring the progress of bug fixing by various graphical/charting facility and status updates, providing reliability bug prediction and bug complexity measurements, and distributing fixes to users/developers.

Defect Tracking System, Sujata Solanke and Prof. Prakash N. Kalavadekar, 2014

For the improvement of software quality now a days Defect Tracking System has been developed. There are various already existing methods like Redmine, Bugzilla etc. which doesn't meet the criteria of perfect defect tracker. This paper is aimed at developing an online defect tracking system useful for applications developed in an organization. The Defect Tracking System (DTS) is a web based solicitation that can be accessed throughout the organization. In this system can be used for sorting defects against an application/module, assigning defects to individuals and tracking the defects to resolution. This solicitation contains features like email notifications, user maintenance, user access control, report generators etc. This paper has been planned to be having the view of distributed architecture, with centralized storage of the database. The system for the storage of the data has been scheduled. Using the paradigms of MS-SQL Server and all the user interfaces has been designed using the ASP.Net technologies. The principles of security and data protecting mechanism have been given a big choice for proper procedure. The solicitation takes care of different modules and their related reports, which are created as per the applicable strategies and principles that are put forwarded by the administrative staff. This system will overcomes all problems of previously existing bug trackers.

3. PROPOSED SYSTEM

A bug vault (a normal programming archive, for putting away points of interest of bugs), assumes a critical part in overseeing programming bugs. Programming bugs are unavoidable and altering bugs is costly in programming improvement. Programming organizations spend more than 45 percent of expense in settling bugs. Extensive programming ventures convey bug vaults (additionally called bug following frameworks) to bolster data gathering and to help engineers to handle bugs. In a bug storehouse, a bug is kept up as a bug report, which records the literary portrayal of recreating the bug and upgrades as indicated by the status of bug settling. A bug storehouse gives an information stage to bolster numerous sorts of assignments on bugs, e.g., shortcoming forecast, bug restriction, and revived bug investigation. In this paper, bug reports in a bug archive are called bug information.

There are two challenges related to bug data that may affect the effective use of bug repositories in software development tasks, namely the large scale and the low quality. On one hand, due to the daily-reported bugs, a large number of new bugs are stored in bug repositories. Taking an open source project, Eclipse, as an example, an average of 30 new bugs are reported to bug repositories per day in 2007; from 2001 to 2010, 333,371 bugs have been reported to Eclipse by over 34,917 developers and users. It is a challenge to manually examine such large-scale bug data in software development. On the other hand, software techniques suffer from the low quality of bug data. Two typical characteristics of low-quality bugs are noise and redundancy. Noisy bugs may mislead related developers while redundant bugs waste the limited time of bug handling.

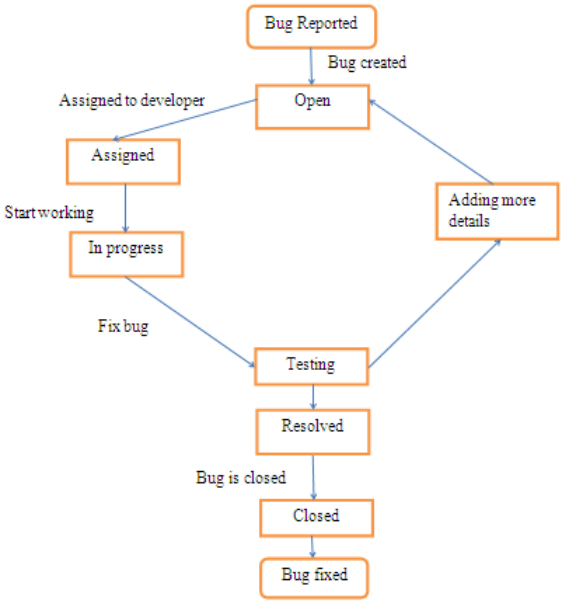
4.K-MEANS CLUSTERING

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the k-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k-means because of the k in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by k-means to classify new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

5. SYSTEM ARCHITECTURE



6. MODULES

Login

This module makes the guest to a member of this website. The guest has to fill up his personal profile through online. After getting a user id and password he can able to login in the website.

New Bug Code, Mark the Bug

A new bug code is created in order to maintain the unique values in the database for a bug. If the user founds a new bug then he can create a new bug code and post it to others. It helps the other users to select the bug to assign for a new bug.

Corrector

The bug corrector can correct the bugs to rectify the errors which are in posted by the testers.

7. CONCLUSION

It is a costly stride of programming support in both work cost and time cost. In this paper, we consolidate highlight determination with occasion choice to decrease the size of bug information sets and in addition enhance the information quality. To decide the request of applying occurrence choice and highlight choice for another bug information set, we extricate characteristics of every bug information set and prepare a prescient model taking into account authentic information sets. We exactly research the information lessening for bug triage in bug vaults of two expansive open source ventures, to be specific Eclipse and Mozilla. Our work gives a way to deal with utilizing procedures on information preparing to frame decreased and top notch bug information in programming advancement and support.

REFERENCES

[1] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst, "Finding bugs in web applications using dynamic test generation and explicit-state model checking," IEEE Softw., vol. 36, no. 4, pp. 474–494, Jul./Aug. 2010.

[2] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," ACM Trans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.

[3] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013. [5] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/

[4] K. Balog, L. Azzopardi, and M. de Rijke, "Formal models for expert finding in enterprise corpora," in Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, Aug. 2006, pp. 43–50.

[5] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.

[6] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," Data Mining Knowl. Discovery, vol. 6, no. 2, pp. 153–172, Apr. 2002.

[7] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, Feb. 2010, pp. 301–310.

[8] V. Bol_on-Canedo, N. S_anchez-Maro~no, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," Knowl. Inform. Syst., vol. 34, no. 3, pp. 483–519, 2013.

[9] V. Cerver_on and F. J. Ferri, "Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule," IEEE Trans. Syst., Man, Cybern., Part B, Cybern., vol. 31, no. 3, pp. 408–413, Jun. 2001.

[10] D.Cubrani_c and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.

[11] Eclipse. (2014). [Online]. Available: http://eclipse.org/

[12] B. Fitzgerald, "The transformation of open source software," MIS Quart., vol. 30, no. 3, pp. 587–598, Sep. 2006.

[13] K. Farahat, A. Ghodsi, M. S. Kamel, "Efficient greedy feature selection for unsupervised learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 285–310, May 2013.

[14] N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, 2nd ed. Boston, MA, USA: PWS Publishing, 1998.
[15] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in Proc. 13th Int. Conf. Mach. Learn., Jul. 1996, pp. 148–156.
[16] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," Knowl. Inform. Syst., vol. 35, no. 2, pp. 249–283, 2013.
[17] Guyon and A. Elisseeff, "An introduction to variable and feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003.
[18] M. Grochowski and N. Jankowski, "Comparison of instance selection algorithms ii, results and comments," in Proc. 7th Int. Conf. Artif. Intell. Softw. Comput., Jun. 2004, pp. 580–585.
[19] Gao, T. M. Khoshgoftaar, and A. Napolitano, "Impact of data sampling on stability of feature selection for software measurement data," in Proc. 23rd IEEE Int. Conf. Tools Artif. Intell., Nov. 2011, pp. 1004–1011.
[20] E. Hassan, "The road ahead for mining software repositories," in Proc. Front. Softw. Maintenance, Sep. 2008, pp. 48–57.

