MANTECH Publications
Recent Trends in Computer Science and Software Technology
Volume 2, Issue 1, January-April 2017

Implementation of an Efficient Bug Tracking System

Shreya Arudkar[1], Amit Pimpalkar[2]
Department of Computer Science & Engineering
Rashtrasant Tukadoji Maharaj Nagpur University
Nagpur, Maharashtra, India
Corresponding Authors: *shreyaarudkar5@gmail.com[1], amit.pimpalkar@raisoni.net[2]*

Abstract

Nowadays IT corporations area unit payment over 45% of their value in fixing code bugs. Historically these bugs area unit fastened by manual assignment to a selected developer; this approach causes an excessive amount of dependency. The new and various approach is that the Bug Tracking System, that fixes the bug and assigns the reportable bug to a developer mechanically in order that it decreases the time and price in manual work. Mix instance selection with feature selection to at the same time cut back information scale on the bug dimension and therefore the word dimension. we have a tendency to propose to use machine learning technique in bug sorting to predict that developer ought to be assigned on the bug, supported its description by applying text categorization.

Keywords: *Data reduction, Instance selection, Feature selection, Bug triage, Machine learning technique.*

I. INTRODUCTION

Automated Bug Tracking System (BTS) has its significance in massive package development comes that manages bug reports and list of developers work on fixing them. BTS has its importance in open supply package development, wherever the team members will be spread round the world. In such distributed comes, the developers and alternative contributors could seldom see one another. And, the BTS is employed not solely to stay track of downside reports and have requests, however it facilitate in coordinative the work among the various developers.

Software bugs will never be evitable and at identical time fixing bugs is pricey in package development. Massive package comes deploy bug repositories to support data assortment and to help developers to handle bugs. A bug repository can contain all report that plays a very important role in managing package bugs. In a very bug repository, a bug is maintained as a bug report that records the whole description of reproducing the bug and updates consistent with the standing of bug fixing. A bug repository offers an information platform to support many sorts of tasks on bugs, e.g., fault prediction, bug localization, and reopened bug analysis.

The bug reports in a very bug repository are referred to as bug knowledge. This bug report is then assigns to a developer, starts to mend the bug. If the assigned developer is unable fix the bug, the bug is migrated to a different developer. The method of assignment a bug report back to Associate in applicable developer is named bug sorting, because the work of BTS is to settle on the suitable developer for fixing bugs, follow the prevailing work to get rid of unfixed bug reports.

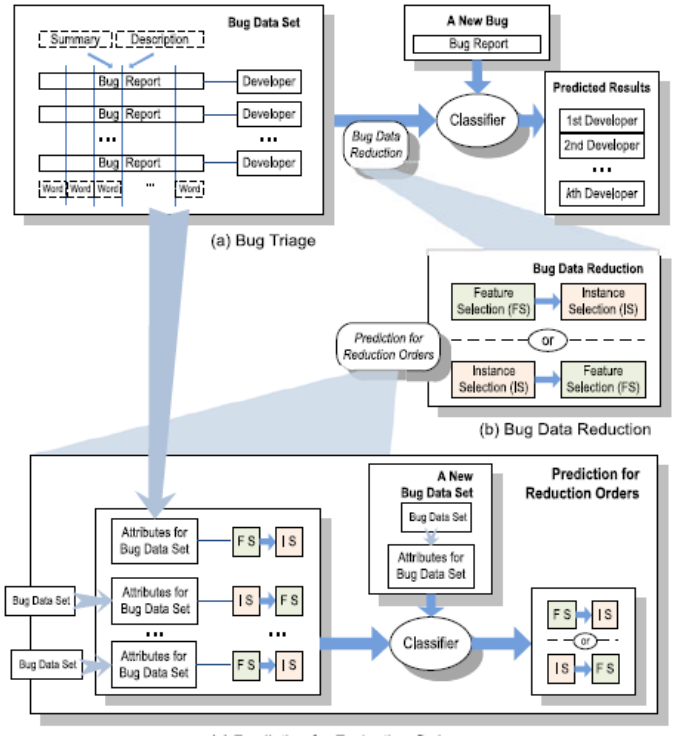

Figure 1. Architecture for Bug Tracking System

II. RELATED WORK

Cubrani planned the matter of automatic bug sorting. The Machine Learning technique, Text Categorization was applied to help in bug sorting by using text categorization. Text categorization could be a technique of mechanically sorting a collection of documents into classes from a predefined set wherever a developer gets expected using the bug's description, for this that they had used supervised machine learning technique using Naive Bayes classifier to predict the proper developer [1,2].

Xuan gave a semi-supervised approach for automatic bug sorting using text classification. This approach combined the Naive Bayes classification approach and expectation maximization to require the advantage of each tagged and unlabelled bug reports. He trained a classifier with a fraction of tagged bug reports. This tagged varied unlabelled bug reports. From the results of, this semi-supervised approach improves the classification accuracy of bug sorting by up to six and it avoids low-quality bugs [3].

When a bug report has been appointed to a selected developer, then if the appointed developer is unable to repair the bug, the appointed developers will forward or transfer the bug to different developer. This method of assignment of bug from developer to developer is named "Bug Tossing" [4]. Additionally, a model of bug agitated is constructed to scale back the quantity of assignment of bug reports. The Markov chains primarily based tossing graph approach was planned to capture the bug tossing history that improved the bug assignment and reduced superfluous tossing steps.

Instance choice and have choice square measure normally used techniques in processing. For a given information set, instance choice is to get a set of relevant instances (i.e., bug reports in bug data) [5] whereas feature choice expects to get a set of relevant options (i.e., words in bug data) [6].

Set of machine learning tools and a probabilistic graph-based model (bug tossing graphs) that square measure highly-accurate predictions, and set the inspiration for future generation of machine learning-based bug assignment. They used methodology like selecting effective classifiers and options, progressive learning,

multi-featured tossing graphs to attain their goal [7].

This technique models the assignment of bugs as Enriched Adaptive Bug Triaging System (EABTS) that was supported actual path model. The graph structure captured the link among developers because the variety of tosses and conjointly captures the proximity exists among developers. Therefore, this graph structure was enriched. The technique was supported ant routing. hymenopter routing is inherently adaptive in nature. Their work gave a sub graph that consists of developers WHO were oftentimes concerned in bug resolution [8].

To investigate the standard of bug information, style questionnaires to developers and users in 3 open supplies comes. supported the analysis of questionnaires, they characterize what makes an honest bug report and train a classifier to spot whether or not the quality of a bug report ought to be improved [13]. Duplicate bug reports weaken the standard of bug information by delaying the price of handling bugs. To discover duplicate bug reports, they style a natural language process approach by matching the execution data.

The existing systems machine learning techniques are ineffective for big project. so P. Bhattacharya et al. [15] projected a way for bug sorting. Goal of this paper is to search out the best set of machine learning techniques to enhance bug assignment accuracy in massive projects. This paper used a comprehensive set of machine learning tools still as a probabilistic graph-based model (bug tossing graphs) that cause highly-accurate predictions, and set the muse for succeeding generation of machine learning-based bug assignment. They used methodology like selecting effective classifiers and options, progressive learning, Multi-featured tossing graphs to attain their goal.

III. PROPOSED SYSTEM

Data Sets: We are going to choose all reports entered into Eclipse's bug tracking system between January one, 2002 and Sep one, 2002. A complete of fifteen,859 reports were elite.

Report Generation: Finally, every report can store a time sealed history of all the changes to its attributes, as well as the

assigned-to. To see a document's category the approach would be to settle on whoever was assigned to the report. We are going to use our observations and experiences with the bug tracking and development procedures within the Eclipse project and devised the subsequent heuristic to see a report's category.

Bug trailing System: we have a tendency to propose to use machine learning techniques to help in bug sorting to predict that developer ought to be assigned on the bug supported its description by applying text categorization, i.e. however the standard of bug information would be improved. Following are the various scenarios for our projected Bug tracking System

- ***Scenario 1:*** In 1st scenario the assigned developer can resolved the report and it'll be tagged by the developer's category no matter who has submit the report or the sort of resolution.

- ***Scenario 2:*** In second scenario, the report may be resolved by somebody apart from the assigned developer, however not by the one that submitted it or nor by the person directly assigned, the report are tagged with the category of the actual developer who marked it resolved. The reasoning is that whoever resolves the report is that the person to whom it ought to be assigned.

- ***Scenario 3:*** during this scenario, if the report is resolved as mounted, no matter whom the resolver was, we have a tendency to assume that this is often the developer who enforced the ?x and label the report with the category of the assigned developer. This rule covers the frequent case wherever Associate in Nursing Eclipse developer files a report, that is then assigned to someone else or a sub-team alias by default, so later implements the ?x himself. Processing Re-write Suggestions Done (Unique Article)

- ***Scenario 4:*** If the report was resolved as non-fixed by the one who submitted it, and who wasn't additionally assigned to that, the report is category labelled with the primary assigned developer or person. Since it would be a feature

or a bug which can later be handled by somebody or when being prompted by a developer for details of his or her setup and find out that there's no bug.

- ***Scenario 5:*** If the submitter resolve the report as non- fixed who the assigned-to developer wasn't, and no-one responded, there's a mistake or the submitter caught the error before anyone began to solve the difficulty, these reports are removed from the training set, because it can't be faithfully labelled.

- ***Scenario 6:*** If no developer may resolve the report then it's marked as non-fixed and therefore the category is assigned by the developer who was the last one that has worked on the report.

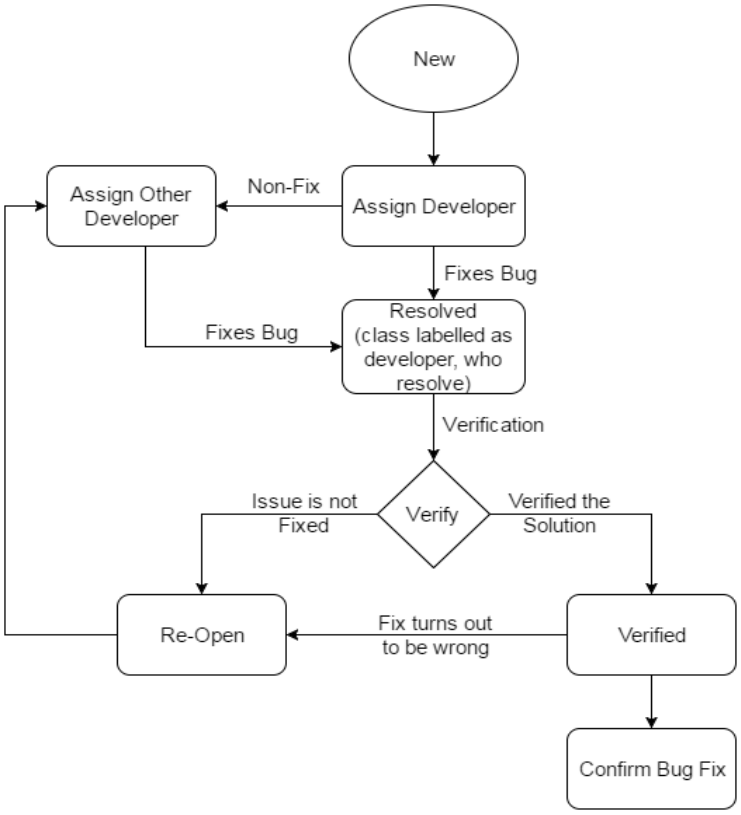

Figure2. Scenarios of Solving Bug

In this paper, we have a tendency to concentrate on the efficient technique to beat drawback of data reduction for bug triage with reducing the size of bug data.

Algorithm: Random Forest algorithm

Input: node M having m1, m2, m3..... variables

1) Grow a forest of the many trees. (R default is 500)

2) Grow every tree on an independent bootstrap sample from the training data.

3) At every node:
    a. Choose m variables randomly out of all M attainable variables (independently for every node).

    b. Realize the most effective split on the chosen m variables.

4) Grow the trees to most depth (classification).

5) Vote/average the trees to induce predictions for brand new data.

IV. BUG PRIORITY

Defect Priority signifies however necessary & urgently it's to repair this defect. In different words Priority suggests that how fast it's to be fixed. The Defect priority status is ready by the project manager supported the client needs & supported Project Priorities the merchandise fixes is finished. Priority is said to planning to resolve the matter. It's a pointer towards the importance of the bug & if High priority is mentioned then the developer must fix it at the earliest. It's associated with technical side of the product. It reflects on however dangerous the bug is for the system and additionally mostly associated with Business or selling facet.

To determine the Defect priority next few points has to be considered:

- Urgency as Business purpose of read to repair the defect
- Impact of the defect on practicality
- Visibility of the defect

- Check if resources area unit offered or to not fix this suggests developer to repair this and Testers to verify the fixes.
- Main constraint in accessibility of time to fix the defect.

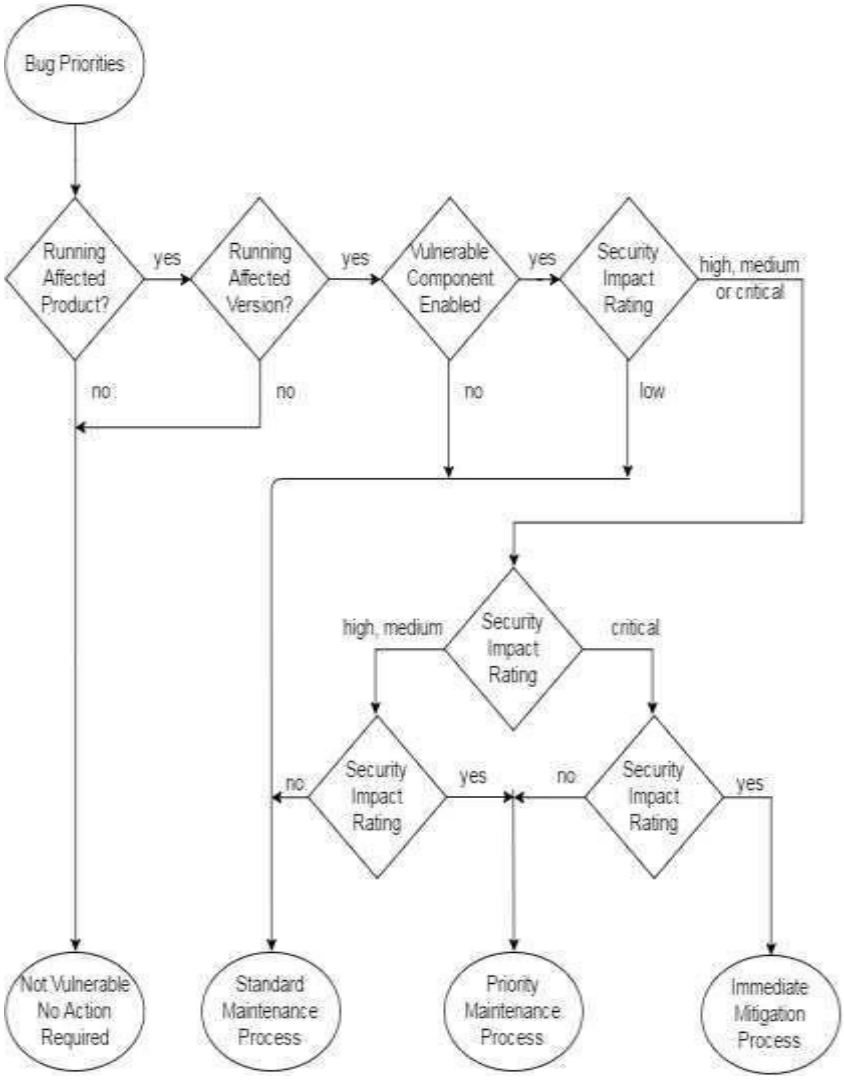

Figure 3. Flowchart of Bug Priorities

V. EXPECTED OUTCOME

Bug sorting is an expensive step of software system maintenance in each labour value and time value. The approach mix feature selection with instance selection to scale back the size of bug data sets also as improve the information quality. to work out the order of applying instance selection and have choice for a replacement bug data set, we tend to extract attributes of every bug data set and train a predictive model supported historical data sets. The work can offer an approach to investing techniques on processing to create reduced and high-quality bug data in software system development and maintenance.

CONCLUSIONS

Data processing techniques like instance selection and have selection square measure used for data reduction. The planned system are often used for any open source comes that generate vast bug data. The techniques of instance selection and feature selection are used to reduce noise and redundancy in bug data sets. The data reduction effectively reduces the bug data set into prime quality bug data which may be used for effective bug triaging. The prediction order is decided by extracting the attributes of the bug data sets. We tend to perform the experiment of the information reduction for bug triage in bug repositories of two giant open sources comes like Eclipse and Mozilla.

Our planned system is predicated on Random Forest that provides an economical approach on processing to scale back the size and supply high-quality bug knowledge in software system development and maintenance. Our system improves the accuracy of tracking system.

REFERENCES

1) Shreya Arudkar and Amit Pimpalkar "Design of an Effective Mechanism for Automated Bug Triage System" Jan 2017.

2) D. Cubrani and G. C. Murphy, "Automatic bug triage using text categorization," in Proc. 16th Int. Conf. Softw. Eng. Know l. Eng., Jun. 2004, pp. 92–97.

3) J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.

4) J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Automatic bug triage

using semi-supervised text classification," in Proc. 22nd Int. Conf. Softw. Eng. Knowl. Eng., Jul. 2010, pp. 209–214.

5) G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with tossing graphs," in Proc. Joint Meeting 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug. 2009, pp. 111–120.

6) 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.

7) I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003.

8) T. Zhang, G. Yang, B. Lee, I. Shin "Role Analysis-based Automatic Bug Triage Algorithm", 2012.

9) V.Akila, Dr.G.Zayaraz, Dr.V.Govindasamy "Effective Bug Triage – A Framework", International Conference on Intelligent Computing, Communication & Convergence, 114 – 120, 2015 .

10) 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.

11) 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.

12) C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

13) 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.

14) T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schr€oter, and C. Weiss, “What makes a good bug report?”, Oct. 2010, IEEE Trans. Softw. Eng.,

15) X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, “An approach to detecting duplicate bug reports using natural language and execution information,” in Proc. 30th Int. Conf. Softw. Eng., May 2008.

16) P. Bhattacharya, L. Neamtiu, C. R. Shelton “Automated, highly-accurate, bug assignment using machine learning and tossing graphs” , 2012.

17) Partha S. Bishnu and Vandana 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.

18) 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.