ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Automatic Bug Triage using Data Reduction & Domain Specific Bug Classification

Nithesh.K.V[1], Savitha.S.K[2]

M.Tech Student, Dept. of CSE, Bangalore Institute of Technology, Bengaluru, India[1]

Assistant Professor, Dept. of CSE, Bangalore Institute of Technology, Bengaluru, India[2]

ABSTRACT: Bugs in the software are impossible to avoid and hence dealing with them takes a lot of time. Great amount of monetary value is exhausted by the companies in handling such bugs these days. Automatic Bug triage is an inevitable step to fix the everyday bugs, which automatically assigns a person to a freshly generated bug in a right way. To assign the bug to the expertise in the specific domain, domain specific bug assortment methods are enforced to lead the bug management. Here, we tend to cover the matter of reducing the data for the sorting, i.e., a way to scale back the size and amend the standard of the data. We work on the feature and instance extraction in the same order to scale back information scale on the word and bug proportion. Our act also renders associate degree method to investing techniques on working to create concentrated and high standard information of bugs in software packages.

KEYWORDS: Bug Triage; Data Reduction; Feature Extraction; Instance Extraction; Domain Specific Bug Classification

I. Introduction

Bug repositories are the software repositories which store the bug details. Software bugs are generated each and every day and fixing them is a big problem and expensive too. Bugs are retained as reports in the repository that consists of the bug's status whether it has been fixed or not and the detailed description of reproducing it. Here, we have considered the bug data as the reports of the bugs in a repository.

The challenges that are associated with it which affects the productive utilization of the repositories are the minus quality & the extensive data. Prominent quantities of raw bugs are generated and are put in the repositories due to the large number of reported bugs every day [1]. It is quite a dare to look into this extensive data in the growth of the project only by manual means. Techniques of the Software system also have to bear from the poor standard of bugs. The distinctive features of such minus standard bugs are redundancy & disturbance. Unnecessary bugs take lot of fixed time to maintain the bugs while the bugs that are disturbing may misguide the developers.

Bug triage is a long method of maintaining these bugs that intends to allot an exact person to solve a bug when it is generated. In this existing system, the fresh bugs are manually sorted by a developer who is an expert. Manual doing it is quite overpriced in consumption of time and lack in correctness too due to the large quantity of bugs that are reported everyday and the deficiency of expertness to handle the bugs [1]. To annul this overpriced expenditure of doing it manually, our present work has offered a machinelike bug management perspective, which enforces domain specific bug methods to anticipate the proper person to solve the bug reports based on the domain.

Nevertheless, the extensive data & its minus quality in the repositories of the bugs stop the designed methods of machinelike triaging of bugs. Software bug data are made by the developers and they are kind of free-form data and it is very important to give a properly organized data [7]. Here, we cover the above mentioned trouble of reducing the data for machinelike triage, i.e., to bring down the data of bug to preserve the task price of users and how to ameliorate the standard of the data to ease the procedure of machinelike triage. Reduction of data for automatic method targets to create a rich-standard and small bug data by getting rid of the words and reports, which are surplus or unnecessary. Here we blend the present methods of feature extraction along with instance extraction to decrease the word and bug proportion at a time. The final output data consists of less number of words and lesser written reports of the bugs than the original bug set.

II. Related work

Bugs that are freshly created are managed by hand of a proficient person where in he/she allots a new bug to a software developer manually [1]. As there are plenty of day-to-day bugs and the deficiency of knowledge of different variety of faults, manual way of doing it is a waste of time & it's even less accurate. Feature and Instance Extraction are applied randomly to reduce the noisy and redundant reports. Correctness of the bug manager will be slightly diminished because of implementing Instance Extraction first which deletes the bug reports that are uninformative. Bugs can be fixed by users, testers or developers [2]. Specialized bugs are maintained by Expert developers and they rectify them whenever the bugs are generated. Searching document repositories of an organisation for experts gives a cost effective solution for the work of expert finding [3]. We introduce some general expert schemes to finding provided a document accumulation that is validated with the use of productive probabilistic patterns. One strategy is to instantly model a skilful cognition grounded on the given documents they are associated with, while another locates documents on the given topic, and then discovers the related proficient. Shaping authentic connections is significant to the execution of Proficient Searching systems. The fundamental closest neighbour classifier bears from the general depot of all prepared instances that are presented. Classification response time slows down when there is a huge database of instances. When instances are noisy, accuracy of the classification may suffer [4]. By removing instances, both problems can be relieved, but the standard used is distinctly assumed to be all effective over domains. We argue against this perspective and propose an algorithm that rivals the existing algorithm. When evaluated on variety of problems, both the algorithms don't systematically outperform each other: consistency is very hard. We need to develop systems that provide insights into class definitions' structures to achieve the good results. We talk about the possibility of such mechanisms and introduce some measures that could be of some use for the data miner. Bug management structures play a cardinal function in affirming coactions among the users and the developers for many projects [5]. To improve the tool support, we have qualitatively & quantitatively examined the problems enquired in a group of large number of the reports from the ECLIPSE & other companies. We placed those problems in categories and examined reaction times and by category of the project. Our consequences prove that the user's role is not only about filing bugs but their ongoing and active involvement is prominent for doing advancement on the reported bugs.

III. Proposed system

A. *Description of the Proposed System:* To avoid the malice of manual procedure, an automatic bug triage has been planned. In this plan, the domain specific bug classification techniques are applied to assign the proper developers for bug reports depending on their domain expertise.

- Users can submit the bug reports in the web portal given.
- When the report is submitted, it automatically gets allotted to a domain expertise or a developer based on the domain of the bugs.
- Developer or Domain expertise can accept the bug reports and solve it or reject it due to his/her time constraint.
- Rejected bug report goes to the next domain expertise in the respective domain list automatically.
- Admin or the expert developer has every right to assign the bug report to any developer manually.

The Proposed system also focuses to increase the information of bug management system in these features, that is to say

- To decrease the size of the proportions of words & bugs simultaneously.
- To ameliorate the bug triage's accuracy.

A combinatory way of applying feature extraction first [8] [9] and then the instance extraction removes words which are uninformative thereby increasing the accuracy of the automatic bug triage.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

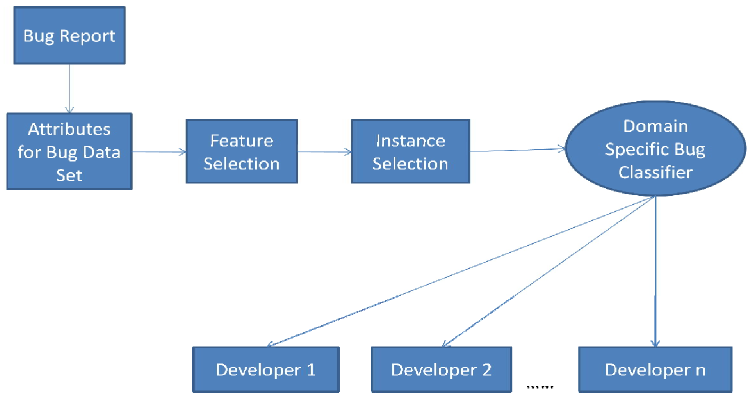


Fig.1. System Architecture

B. *Advantages of the Proposed System:*

- Automatic bug sorting or triaging reduces the time cost in manual approach.
- Domain specific bug classification techniques are applied so that the bug reports are mechanically assigned to the developer of the specific domain depending on the domain of the bug report.
- It reduces the dimensions scales of bugs and words at a time.
- It ameliorates the accuracy of the automatic bug triage as we apply the feature selection and instance selection sequentially.

IV. PSEUDO CODE

```
Pseudo code for Instance and Feature Extraction
Intake set: Preparing band T with b bug reports & w words,
            data simplification order FE to IE
            last count is $w_f$ of words,
            last count is $b_i$ of reports,
Output set: Diluted set $T_{fi}$ for automatic bug manager
            1) use FE to w words of T and calculate target values
            for the words;
            2) choose the top $w_f$ words of T and get a prepared
            band $T_f$;
            3) use IE to $b_i$ number of reports of $T_f$;
            4) finish IE when the number of total reports is equal to
            or less than bi and get the final prepared band $T_{fi}$.
```

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

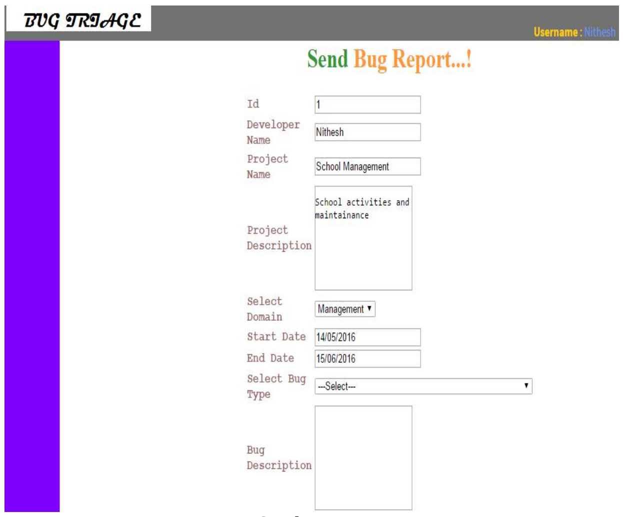
International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

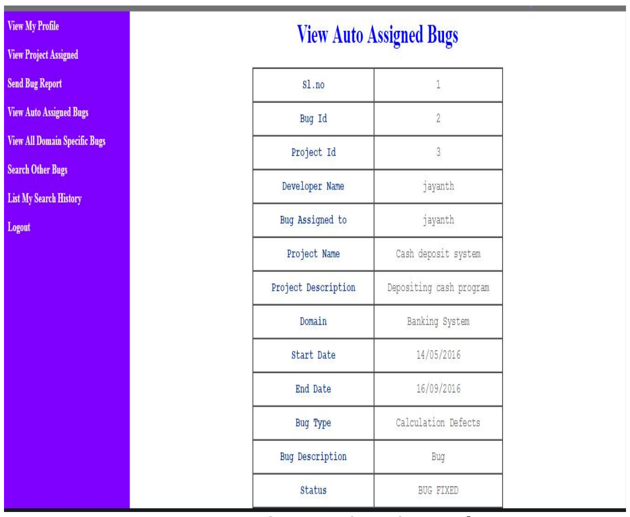
V. RESULTS

The results of the proposed system are shown in the following figures.



Id	1
Developer Name	Nithesh
Project Name	School Management
Project Description	School activities and maintainance
Select Domain	Management
Start Date	14/05/2016
End Date	15/06/2016
Select Bug Type	--Select--
Bug Description	

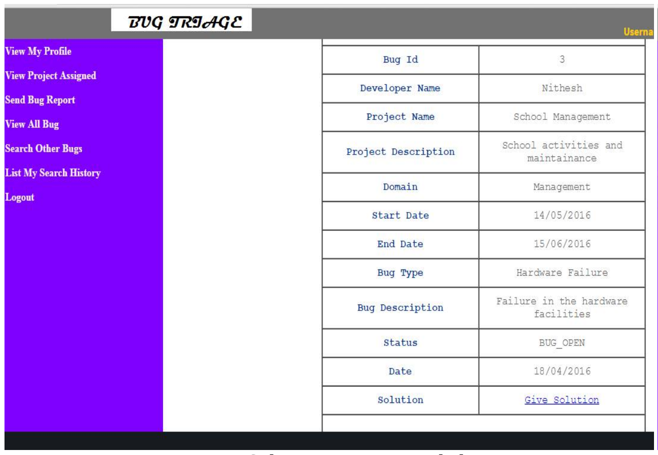
Fig.2. Send Bug Report



Sl.no	1
Bug Id	2
Project Id	3
Developer Name	jayanth
Bug Assigned to	jayanth
Project Name	Cash deposit system
Project Description	Depositing cash program
Domain	Banking System
Start Date	14/05/2016
End Date	16/09/2016
Bug Type	Calculation Defects
Bug Description	Bug
Status	BUG FIXED

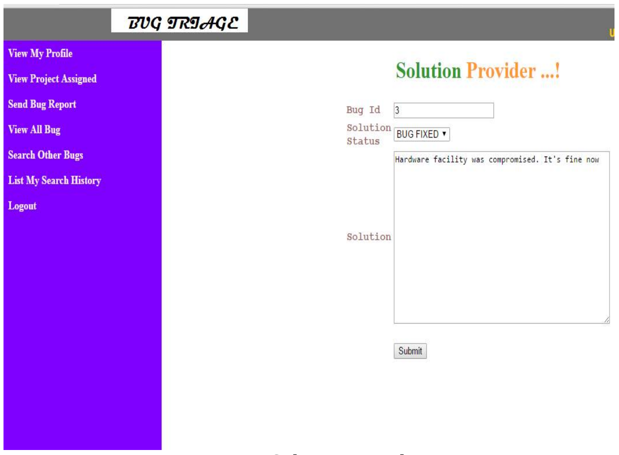
Fig.3. View Auto Assigned Bugs

In Fig.2, the page to send a bug report is shown and all the details that are necessary are entered and the bug report is filed accordingly. Then the bugs are auto assigned to the experts based on the domain specified. Fig.3 shows the auto assigned of the bugs portal of the respective user. Users can either accept or reject the bug report. If users reject it, then the bug is automatically assigned to the next developer in line based on the ranking system.



Bug Id	3
Developer Name	Nithesh
Project Name	School Management
Project Description	School activities and maintainance
Domain	Management
Start Date	14/05/2016
End Date	15/06/2016
Bug Type	Hardware Failure
Bug Description	Failure in the hardware facilities
Status	BUG_OPEN
Date	18/04/2016
Solution	Give Solution

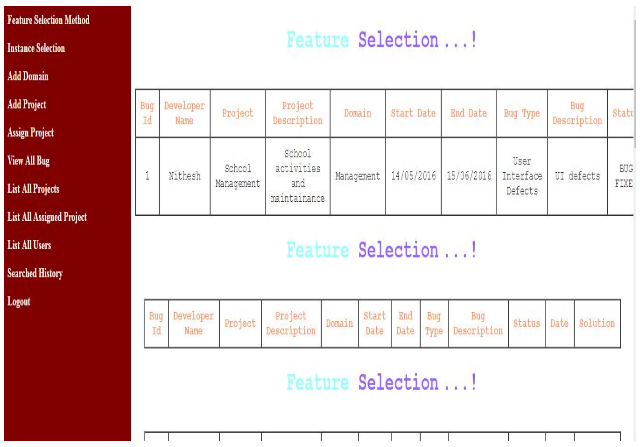
Fig.4. Solution not yet provided



Bug Id	3
Solution Status	BUG FIXED
Solution	Hardware facility was compromised. It's fine now

Fig.5. Solution Provider

In Fig. 4, Users portal is shown where in when users check the assigned bugs, the pending solution is displayed. Then users can click on 'Give Solution' to provide solution to the bug report. In Fig. 5, users can provide the solution status i.e. fixed, pending etc and even solution of the bug report and submit it.



Feature Selection ...!

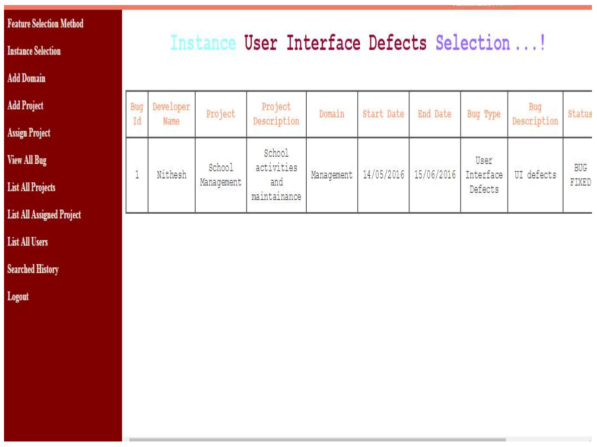
Bug Id	Developer Name	Project	Project Description	Domain	Start Date	End Date	Bug Type	Bug Description	Statu
1	Nithesh	School Management	School activities and maintainance	Management	14/05/2016	15/06/2016	User Interface Defects	UI defects	BUG FIXE

Feature Selection ...!

Bug Id	Developer Name	Project	Project Description	Domain	Start Date	End Date	Bug Type	Bug Description	Status	Date	Solution

Feature Selection ...!

Fig.6. Feature Selection of the bugs



Instance User Interface Defects Selection ...!

Bug Id	Developer Name	Project	Project Description	Domain	Start Date	End Date	Bug Type	Bug Description	Status
1	Nithesh	School Management	School activities and maintainance	Management	14/05/2016	15/06/2016	User Interface Defects	UI defects	BUG FIXED

Fig.7. Instance Selection of the bugs

In Fig. 6 & 7, it's shown that users can view all the bug reports submitted. They can view all the bugs based on the features or separate bugs based on the specific instances like User interface defects.

VI. CONCLUSION AND FUTURE WORK

Managements of bugs are pricey in terms of toil and consumption of time. Here, an automatic bug triage has been made using domain specific bug classification technique to reduce the manual work there by saving the time cost. We have also merged the important methods of data selection to bring down surmount of sets of bug information and amend the quality of such bugs. We have provided a different set about to hold proficiencies on processing of the data to create rich-quality & thinned out data of the bugs. In the succeeding work, we want to make the set of bug data a best quality one by ameliorating the consequences of reduction of data in automatic bug management and we want to take on a domain-specific task of the software.

REFERENCES

1. He Jiang, Zhilei Ren, and Yan Hu, "Towards effective bug triage with Software data reduction techniques" in IEEE. Trans. Knowledge and Data Eng., Vol.27, No.1, January 2015.
2. G. C. Murphy, L. Hiew, and J. Anvik, "Who should fix this bug?" in Proc. 28th Int. Conf. Softw. Eng., pp. 361–370, May 2006.
3. M. de Rijke, Azzopardi, and K. Balog, L, "Formal models for expert finding in enterprise corpora," in Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval, pp. 43–50, Aug. 2006.
4. C. Mellish & H. Brighton, "Advances in instance selection for instance-based learning algorithms," in IEEE. Conf. Data Mining Knowl. Discovery, vol. 6, no. 2, pp. 153–172, Apr. 2002.
5. T. Zimmermann, S. Breu, R. Premraj, & J. Sillito, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, pp. 301–310, Feb. 2010.
6. S. Artzi, M. D. Ernst, A. Kie_zun, F. Tip, A. Paradkar, D. Dig and J. Dolby, "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.
7. Bugzilla, (2014). [Online]. Available: http://bugzilla.org/
8. V. Bol_on-Canedo, N. S_anchez-Maro~no, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," in IEEE. Conf. Knowl. Inform. Syst., vol. 34, no. 3, pp. 483–519, 2013.
9. A. K. Farahat, A. Ghodsi, M. S. Kamel, "Efficient greedy feature selection for unsupervised learning," in IEEE. Conf. Knowl. Inform. Syst., vol. 35, no. 2, pp. 285–310, May 2013.
10. Eclipse. (2014). [Online]. Available: http://eclipse.org/
11. A. K. Farahat, A. Ghodsi, M. S. Kamel, "Efficient greedy feature selection for unsupervised learning," in IEEE. Conf. Knowl. Inform. Syst., vol. 35, no. 2, pp. 285–310, May 2013.
12. 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.
13. 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.