International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 9 (2013), pp. 947-952
© International Research Publications House
http://www. irphouse.com /ijict.htm

Prediction of Top Crashes in Software Systems

Taufeeq Ahmed. K.R

Dept. of CSE, JNTUA College of Engineering (Autonomous)
Anantapuram, India.

Abstract

Now a days there are many software systems emerging most of them instantly reports failure back to the vendor with the intention that developers may have a glance over the most encountering trouble from the software system. However, this consumes time to access those failures which are frequently reported. A crash caused of failure process reported in huge way called Top Crashes and this crash leads the application to failure if not treated. Therefore, prediction of “top crashes" enhances the quality of the software product. Here a machine learner is been featured with all the relevant training of top crashes from the past release. Moreover, this process able to get a quick way out for the most important crashes with enhanced user experience feature and the maintenance efforts over the application. This paper which with a technique to progress the perfectness over the prediction and able to label the defects automatically. Hence, feature like labelling of defects into a category or even in modules which guides the developer to overcome those identified defects over the most defect prone areas and individual developer can easily focus and work on to crack them.

1. Introduction

These days, software systems large in number are brought up in market with the feature of automated problem reporting. Precisely when the trouble encounters, the system reports to the individual with the details of respective problem. An instance of automated problem reporting consider the popular Firefox Internet browser and then with that software system achieve the required data statistics, which are observed when the runtime or Operating system happened to encounter an unrecoverable failure which is nothing but a "crash" and there the software browser perform the required process so as to terminate the task. Moreover, the individual gets a separate "talkback" process which allows the request of individual to submit that crash report which provides the overview of crash occurrence to the Firefox developers (Fig.1).Therefore,every crash report which contains the occurred crash point i.e. program location and thus these crashes which possess the same crash point are then assumed to be the similar one[6].Additionally, the crasheswhich are reported with information related to the crash encountered, with user comments, hardware and software configuration and even the thread stack traces.

Perhaps the figure of crash reports when submitted can be more in number.Every day, the users of Firefox happen to submit crash reports in thousands.Alesseramount of crashes which results in huge figure of crash reports are received called top crashes. Moreover, to detect the top crashesthere an individual need to wait and see until amplein number crash reports are reported and thenin turn it hint at the individual to go through various crashes already,thus this process to fix it might lead tolikely loss of data and frustration. Precisely the main objective behind this paper is to govern whether a crash is a top crash at the head phase it encounters. Therefore, such sort of predictable process can be used to implement so as to find the top crashes at the earlier state of progress. Hence, this may then allow the individual developer to points on top crashes early.

The task to source the top and bottom stack traces provided along with method signatures and these signatures which are delivered to machine learner so that this system instantly divide a crash concise by a new received crash report as common(a top crash) or maybe then rare(a bottom crash).

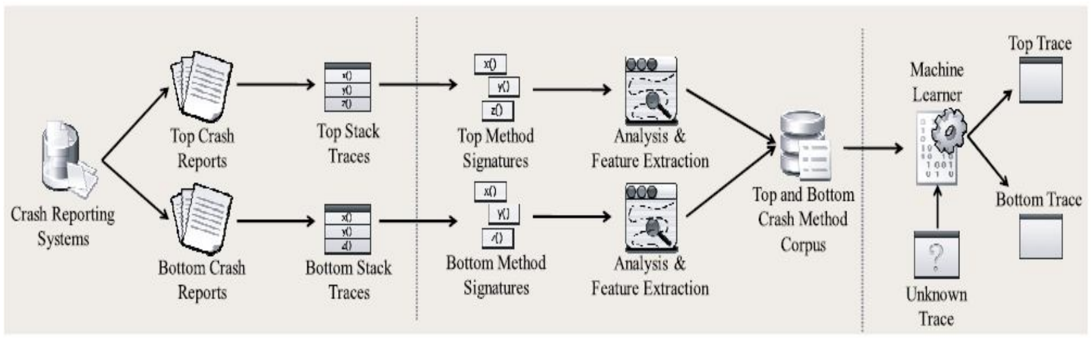

Figure 1: Approach Overview.

Thus these processes motivate for addition of features to the need toincrease the accuracy of the prediction. Therethe aspect rises to propose a machine learner approach which automatically labels those crashes so that it may be easy for the developer to focus over the certain modules in the code and allows the managers to plane their resources on the modules where the top crashes occur.

2. Related Work

These Automatic crash reporting facilities, seems to be integrated over the commercial software. Anyhow the process of analysis relevant to crash reports remains as boring taskwith time taking if carried out manually.The cause behind this drawback is that raw data they contain which certainly does not respond to human research. Perhaps the process which resembles likely near related to failure clustering, an approach introduced by Podgurski et al,[5].Hence, that approach which supports features of selection, clustering along with that multivariate visualization to the respective failures which are affected in similar causes.

Previously, a couple of research group,[4] independently proposed improving clustering accuracy by the automatic fault localization. Therefore entire failure clustering technique is post-mortem analysis, and they even provide with ideas for what sort of utmost mutual failures in the collected reports are.

Perhaps the research in this sort of firm is categorized as profile-based [20], program-based and the last one evidence-based [2] methods. Hence it has been positively implemented so as to predict branch frequency [3].In [2], Calder et al. then proposed an evidence based method which usually states those drawbacks of the program based method while holding the benefits. Moreover, they implemented this method to predict the branch frequency. However, here predicting the crash frequency is been considered from different branch or even with path frequency.Additionally, usage of the social network metrics in this sort of predictions, which donated better quality of accuracy in prediction.

3. Proposed System

The current research which briefly explain the two important objectives and solution for the features .Hence the two objectives are Improving the accuracy of prediction and Automatic Labelling of crash through analysis. Thus the proposed features which have not been considered in [1] for improving the accuracy of prediction. For labelling of crash the current proposed concept is a machine learning approach which is based on neural network and this neural network is against the features and the functionality in which crash is fixed.

For any defects which are categorized as top crash, they are classified using neural network to the functionality. Based on sorting of all top crashes they are summarized and provided with the functional area in which most crashes occurred.

4. Proposed Security Mechanism

The following are considered in the proposed work and the tasks of work which are performed are described in a precise manner.

4.1 Improving the accuracy of Prediction

In the paper work [1], they have not considered environmental factors like the execution environment, OS virtual memory snap, network bandwidth available etc. But most of the crashes occur due to this kind of problems. So for training of the machine learning type then consider the following additional features,

Additional Features

Features	Description
Virtual memory available	Availability of virtual memory in terms of snap
Network bandwidth	The current bandwidth used at the time of crash,
No of instances	Number of instance of application running
CPU usage	CPU usage of the application

For all crashes occurred during the alpha and beta testing these parameters are also gathered and the machine model is trained. Certainly considering these environmental factors, the accuracy of prediction increases. Thus the above features which can then able to improve the accuracy with the respective consideration of factors.

4.2 Auto labeling of crashes

The software can be split into different functional groups. Each functional group can consist of one or more modules. For each defect identified in the alpha and beta testing process, the functional area in which the crash belongs is found by developer. Based on this dataset the crash features with the functional area identified, a feed forward neural network is trained. The input is the features and the output is the functional area of crash. Any defects occurring during the testing and real time usage scenarios, these defect features are extracted and passed to the neural network to identify the functional area in which the crash belongs.

The advantage of this auto labelling is enormous. It allows managers to plan their resources on the functional area in which the top crashes occur. Indirectly defects grouped in similar functional area may have identical source. So it helps the developer to categorize the related crashes together and analyse it.

5. Performance Analysis

In the current work implemented with additional features for identifying top crashes and measured the performance of it against the approach proposed in [1].The need to find that our approach has 5% more accuracy in identifying the top crashes. Therefore, the process measure against different datasets and the performance improvement is consistent.

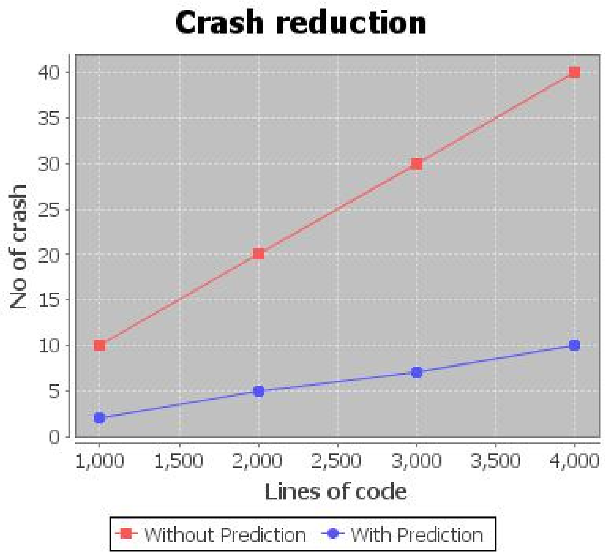

Figure 2: Comparison graph.

6. Conclusion

Now the developers don’t find any reason to worry either to hang up for the crash reports to occur. Fortunately, now it’s time to learn from the past crash reports and then exercise the automatic and effective aspect of process with this sort of prediction. Moreover a new crash report differs from many of the earlier crash one and if it seemssimilar thenseparated.Perhaps with this sort of automatic classification of incoming crash reports, where lets the individual developer to get an instant fix for those most pressing problems and this could enhance the quality of the software with suitable stability and improved users experience. Therefore, this sort of approach with complete automated and ease implementation aspect for any kind of application system where the respective crash data are together in a main significant store.

References

[1] Which crash should fix first? Dongsun Kim IEEE transactions on software engineering MAY 2011.

[2] B. Calder, D. Grunwald, M. Jones, D. Lindsay, J. Martin, M.Mozer, and B. Zorn, “Evidence-Based Static Branch Prediction Using Machine Learning,” ACM Trans. Software Eng. And Methodology, vol. 19, no. 1, pp. 188-222, 1997.

[3] J.A. Fisher and S.M. Freudenberger, “Predicting Conditional Branch Directions from Previous Runs of a Program,” Proc. Fifth Int’l Conf. Architectural Support for Programming Languages and Operating Systems, pp. 85-95, 1992.

[4] C. Liu and J.W. Han, “Failure Proximity: A Fault Localization-Based Approach,” Proc. 14th ACM SIGSOFT Int’l Symp. Foundations of Software Eng., pp. 46-56, 2006.

[5] A. Podgurski, D. Leon, P.A. Francis, W. Masri, M. Minch, J. Sun, and B. Wang, “Automated Support for Classifying Software Failure Reports,” Proc. 25th Int’l Conf. Software Eng., pp. 465-475, 2003.

[6] A. Zeller, “Isolating Cause-Effect Chains from Computer Programs,”Proc. 10th ACM SIGSOFT Symp. Foundations of Software Eng., pp. 1-10, 2002.