A survey on analysing the crash reports of software applications

Asha Ramaraddi Belahunashi[1], Pushpalatha M N[2]

[1]Student, Dept. of ISE, M S Ramaiah institute of technology, Karnataka, India
[2]Asst. professor, Dept. of ISE, M S Ramaiah institute of technology, Karnataka, India

---***---

Abstract - *Software crash is a serious problem in production environment. , when it crashes the crash report is generated and it is sent to developers to debug based on user permission. These crash reports are received by error reporting systems for example windows has windows error reporting system to efficiently handle the crash reports. These crash reports are stored into set of buckets. These buckets may contain duplicate crash reports which are produced by the same bug. The information stored in bucket helps the developer to prioritize the bugs to be fixed. If bucket contains duplicate crash reports it takes developer more time to fix bug and it decreases the efficiency of the bucketing system. Hence we reviewed some existing methods to analyse the crash reports and some methods to group them.*

Key Words: ***Crash reports, Bucketing, Mozilla error reporting system, Stack traces, Windows error reporting system.***

1. INTRODUCTION

Crashes in software are the more sever facts in software bugs. crashes are given top priority to be fixed. To overcome this problem many crash reporting system like windows error reporting system[5],Apple crash reporting system[1],and Mozilla crash reporting system[14] are developed. This error reporting system collects crash reports from end user during the time of crash. Software development team spends more time and resource on testing the software before releasing it. But the software still contains bugs. These bugs will cause the crash. When crash occurs the crash reports are sent to error reporting system like servers. Example WER servers in windows error reporting system [5]. These servers organize the crash reports into multiple buckets and automatically convert them into bug reports. The bug reports finally sent to software developer to fix it [21] as shown in the fig 1.

These crashes are due to executing invalid machine instructions some other causes are incorrect address in program counter, buffer overflow and triggering the unhandled exception. These crash reports contain information like application name, application version, application build date, module name and version, module build date and module offset which is crashed and its call stack traces. These information helps the developer to determine the reason for crash[5,17].In many cases, more number of crash reports are generated. In such cases crash reports generated by the same bug and they are called duplicate crash reports. So it is important to make theses duplicate crash reports to organize in one group it reduces the debugging time and effects of the developers.

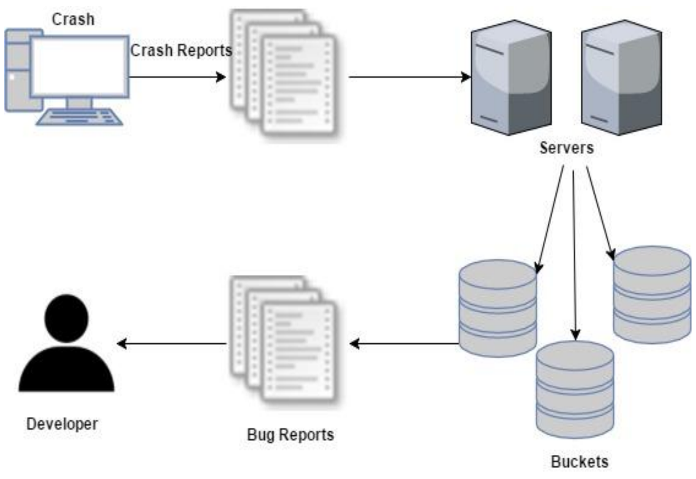

Fig -1 :An overview of crash reporting system

Crash reports in windows error reporting are grouped according to buckets. These buckets contain crash reports which are caused by the same bug. Many methods are used to generate the buckets [5].Based on the number of crash reports stored in each bucket bug fixing efforts are prioritized by the developers. A bucket with large number of crash reports are investigated with high priority compared to bucket with small number of crash reports. It is common that crashes generated by one bug is spread to more than one bucket(the "Second bucket problem")[5].One more problem in WER is only one or few crash reports are present in buckets(The "long tail" problem). These bucketing problem decreases the effectiveness of prioritizing the bugs to be fixed and problem diagnosis.

In recent years, methods for determining the duplicate crash reports are proposed by many researchers. Liu and Han[11] proposed R-Proximity, in which two failed crash traces are similar if they have same fault locations which are determined by fault localization method. Bartz et al[3] proposed a method which uses a call stack similarity to determine duplicate crash reports. These methods requires some parameters to be tuned and automatic learning of these optimal values of the parameters requires more computational cost and it is difficult to implement. Lohman et al [12] proposed a method for quick identification of the duplicate crash reports. In this method the formation of similarity matrix is different. Modani et al [13] proposed a method in which known crash problems are determined based on call stack similarities. In this method they proposed two similarity matrics, top-k indexing and inverted index. The crashes are similar if the top-k frames are same in both call stacks. In inverted indexing if more function are similar between two stacks then they have higher similarity. The crash reports organized by bug tracking system contain information about bug and they are treated as text documents. Some researchers proposed methods for information retrieval techniques to determine the textual similarity between the two crash reports[16].Wang et al[19] combined both textual similarity and execution stack trace similarity to determine the duplicate crash reports. Accuracies in their methods are found around 40% to 60%.Debug advisor[2] is one more recommended system which helps to determine duplicate crash reports and it also provides information which is use full for debugging. It also use call stack similarity to find the similarity of crash reports.

This document is template. We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace(copy-paste) the content with your own material. Number the reference items consecutively in square brackets (e.g. [1]). However the authors name can be used along with the reference number in the running text. The order of reference in the running text should match with the list of references at the end of the paper.

2. BACKGROUND

2.1Some of the existing methods to analyse the crash reports and methods to fix the crash

2.1.1 Measuring the similarity in bug reports

Bogdan Dit[6] proposed a method for measuring textual coherence of user comments in the bug reports. These textual coherence in the user comments may affect the comprehensibility of bug reports hence it is very important to measure the textual coherence. Latent Semantic Analysis[LSA][7,9], is a statistical method for determining the meanings of words in natural language. This method is used to measure the textual coherence of natural language texts. This method also helps to extract the information in stack traces and source code and helps in the automatic assessment of bug reports.

2.1.2 Debugging large crash reports

Windows error reporting system collects the billions of crash reports and it automatically classifies the errors into buckets[8]. The software components grow to large number in the single system. With large number of components, it is very difficult to isolate the main cause of the crash. WER uses a progressive approach to collect crash data and helps the developers to collect detailed information when required. WER uses error statistics as a tool in debugging; this helps developers to isolate the bugs that could be found at smaller scale.

2.1.3 Classifying the crash reports to fix the bugs

Tejinder Dhaliwal [10] proposed Two-Level grouping approach of crash reports. This approach is efficient because it compares only top method signature of the stack traces. In this method crash-type may contain crash-reports caused by multiple bugs. Hence the author suggested two level grouping approaches. In first level grouping it clusters the crash reports based on the top method signature of the stack traces to form crash-types. The subgroups within a crash type create the second level grouping and it helps developers to analyse and file bugs. This method uses Lowenstein distance [20] to determine the similarity between stack traces.

2.1.4 Crash analysis in windows

Analysis of crash or data failure is important for system designers and developers to improve operating system dependability [4]. Most of the operating systems crashes are due to poorly written device driver's codes. The system designers must analyse the data and it helps to understand the root failure caused in the system and helps to build more stable and resilient system. Analysing the failure data or crash reports helps to improve the quality of service.

2.1.5 Locating faulty function based on crash stack information in crash reports

Software often crashes. When crash happens, the crash report with is sent to the development team. Software development team may receive hundreds of stack traces from all deployment sites and many stack traces may be due to same problem. It takes longer duration of time for developer to analyse each traces. Hence Rongxin Wu [15] proposed a crash Locator, a method used to automatically locate the faulty function by generating approximate crash traces by expanding the crash traces. After obtaining the complete stack trace the cyclometric complexity, Functional Frequency, Inverse Bucket Frequency, Average Distance to crash point values are calculated and obtain the overall score and based on the obtained score the functions are ranked and this information is useful for the developers for fixing the problems.

2.1.6 Refresh: a tool used for capturing and reproducing crash reports

Many programs have hidden bugs that cause the program to fail. To fix this problem, it will be difficult to reproduce the failure consistently. Reproducing a failure will be more difficult and takes longer duration of time, especially when the failure is discovered by a user in a deployed application. It is difficult to find and eliminate a software failure, and especially to verify a solution, without the ability to consistently reproduce the failure. S. Artzi [18] proposed a method ReCrash which has two phases. First phase is monitoring phase in which ReCrash maintains a shadow call-stack containing partial state of the arguments to the methods on the original call-stack. When the program fails (i.e., crashes), ReCrash serializes the shadow stack contents, including all heap objects referred to from the shadow stack. Second phase is Test Case Generation ReCrash generates candidate tests by calling methods from the de-serialized shadow call stack. Each test executes the original method using the de-serialized receiver and arguments (stored at the time of the crash). ReCrash outputs multiple tests to create a better view of the failure for the developer.

3. CONCLUSIONS

This paper reviewed the existing methods to analyse the crash reports and methods to group the crash reports to fix the bug. For analysing the crash reports more works like Crash reporting system, Replication of crashes, predicting the modules which are prone and statistical debugging are reviewed. This method helps to reduce the debugging effort and time of the developers and also helps to prioritize which bugs needs to be fixed first.

ACKNOWLEDGEMENT

I thank all faculty members of Department of Information Science, MSRIT and also thanks to the management of MSRIT, Bengaluru for providing the academic and research atmosphere at the institute.

REFERENCES

[1] Apple, “Technical Note TN2123: Crash Reporter”, 2010, available at http://developer.apple.com/library/mac/#technotes/tn2004/tn2123.html.

[2] B. Ashok, J. Joy, H. Liang, S. Rajamani, G. Srinivasa, and V. Vangala, DebugAdvisor: A Recommender System for Debugging, in Proc. ESEC/FSE'09, Amsterdam, The Netherlands, August 2009. pp. 373-382.

[3] K. Bartz, J. W. Stokes, J. C. Platt, R. Kivett, D. Grant, S. Calinoiu, and G. Loihle, "Finding similar failures using call stack similarity," in Proceedings of the Third conference on Tackling computer systems problems with machine learning techniques. San Diego, California: USENIX Association, 2008.

[4] A. Ganapathi, V. Ganapathi, and D. Patterson, "Windows XP kernel crash analysis," in Proceedings of the 20th conference on Large Installation System Administration. Washington, DC, USENIX Association, 2006, pp. 12-12.

[5] K. Glerum, K. Kinshumann, S. Greenberg, G. Aul, V. Orgovan, G. Nichols, D. Grant, G. Loihle, and G. Hunt, "Debugging in the (very) large: ten years of implementation and experience," in Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. Big Sky, Montana, USA: ACM, 2009, pp. 103-116.

[6] Bogdan Dit, Denys Poshyvanyk, Andrian Marcus ,” Measuring the Semantic Similarity of Comments in Bug Reports” , Department of Computer Science Wayne State University Detroit Michigan 48202.

[7] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R., "Indexing by Latent Semantic Analysis", Journal of the American Society for Information Science, vol. 41, 1990, pp. 391-407.

[8] Kirk Glerum, Kinshuman Kinshumann, Steve Greenberg, Gabriel Aul, Vince Orgovan, Greg Nichols, David Grant, Gretchen Loihle, and Galen Hunt, “ Debugging in the (Very) Large: Ten Years of Implementation and Experience” , Microsoft Corporation One Microsoft Way Redmond, WA 98052.

[9] Dumais, S. T., "Improving the retrieval of information from external sources", Behavior Research Methods, Instruments, and Computers, vol. 23, no. 2, 1991, pp. 229 - 236.

[10] Tejinder Dhaliwal, Foutse Khomh, Ying Zou, “Classifying Field Crash Reports for Fixing Bugs : A Case Study of Mozilla Firefox” , Dept. of Electrical and Computer Engineering, Queen’s University, Kingston

[11] C. Liu and J. Han, Failure Proximity: A Fault Localization-Based Approach, in Proc FSE'06, Portland, OR, 2006, pp. 286-295.

[12] G. Lohman, J. Champlin, and P. Sohn. 2005. Quickly Finding Known Software Problems via Automated Symptom Matching. In Proceedingsof the Second International Conference on Automatic Computing (ICAC '05). IEEE Computer Society, Washington, DC, USA, 101-110.

[13] N. Modani, R. Gupta, G. Lohman, T. Syeda-Mahmood, and L. Mignet, Automatically identifying known software problems. In ICDE Workshops, pages 433–441, 2007.

[14] Mozilla, "Crash Stats", 2010, http://crash-stats.mozilla.com.

[15] Rongxin Wu, Hongyu Zhang, Shing-Chi Cheung, and Sunghun Kim, “CrashLocator: Locating Crashing Faults Based on Crash Stacks” , Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China

[16] P. Runeson, M. Alexandersson, and O. Nyholm, Detection of Duplicate Defect Reports Using Natural Language Processing, in Proc. ICSE 2007, Minneapolis,USA, May 2007. pp. 499-510.

[17] A. Schröter, N. Bettenburg, and R. Premraj, “Do stack traces help developers fix bugs?”, in Proc. MSR 2010, Cape Town, South Africa, May 2010. pp. 118–121.

[18] Shay Artzi , Sunghun Kim , Michael D. Ernst , "ReCrashJ: a Tool for Capturing and Reproducing Program Crashes in Deployed Applications"

[19] 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. ICSE'08, Leipzig, Germany, 2008, pp. 461-470.

[20] J. B. Kruskal. —An Overview of Sequence Comparison: Time Warps, String Edits, and Macromolecules, SIAM Review.Vol. 25, No. 2 (Apr., 1983), pp. 201-237

[21] Yingnong Dang, Rongxin Wu, Hongyu Zhang, Dongmei Zhang, and Peter Nobel. Rebucket: a method for clustering duplicate crash re-ports based on call stack similarity. In Proceedings of the 34th Inter-national Conference on Software Engineering, pages 1084–1093. IEEE Press, 2012.

BIOGRAPHIES

Asha Ramaraddi Belahunashi received B.E(CSE) from Gogte Institute of Technology, Belgaum. She is currently Pursuing M.Tech in MSRIT affiliated to VTU, Belgaum. Her research interests include data mining and cloud computing.

Mrs. Pushpalatha M N, is an assistant professor in Department of Information Science and Engineering at M S Ramaiah Institute of Technology, Bangalore-54. Her areas of interest is Software Engineering and data mining.She completed M.Tech in in computer Science and Engineering from M S Ramaiah Institute of Technology, affiliated to VTU.