A Comparative Study of Various Fault Localization Methodologies

Abhilash Sharma
Assistant Professor
CSE Department
RIMT – IET, Mandi Gobindgarh, Punjab, INDIA
abhilash583@yahoo.com

Nidhi Bhatla
Assistant Professor
CSE Department
RIMT – IET, Mandi Gobindgarh, Punjab, INDIA
engineernidhi@yahoo.com

Abstract – Fault in terms of Software Engineering is referred to as a bug, an error or a problem which causes a program to crash or generate invalid results. The problem is caused by either incorrect logic or insufficient logic. Fault localization technique plays a very important role in order to determine such errors and to attain the correct outputs. The objective of this work is to assess different fault localization techniques which are in existence. Thus, this can be favorable to the software tester for fault recognition in the program.

Keywords – Fault; Bug; Testing; N - Gram; Spectrum; Traces.

1. INTRODUCTION

In the software sector, testing is basically carried out to discover the differences between existing and required conditions and to examine the features of the software item. The Software Fault, also known as Software Defect, is a hidden programming error. It is a flaw in the coding that may cause software to behave not in the proposed way and may result in error and hence software failure. Most faults are due to human errors in source code or its design.

A program is said to be faulty when it incorporates a large number of bugs, which affects program functionality and cause erroneous outputs. The larger, more complex a program, the higher the probability of it containing bugs. The problems or faults in the program can also be due to various other reasons. Different sources of such problems are depicted in Figure 1. Generally, test cases are executed in order to perform testing but it is a very time consuming process.

To overcome with the former problem, various fault localization techniques are employed. Fault Localization is a way of finding exactly the location of bugs in the program. Our work endeavors to exhibit various fault localization methodologies which can be valuable to the software tester in many ways.

2. MODUS OPERANDI

It is always challenging for testers to effectively and efficiently remove bugs, while not unwittingly introducing new ones at the same time.This work has been carried out with an aim to analyze various fault localization techniques that can be beneficial for the software tester to identify the suspicious code that may contain program bugs. For this purpose, different publications, research works, reviews and journals have been studied.

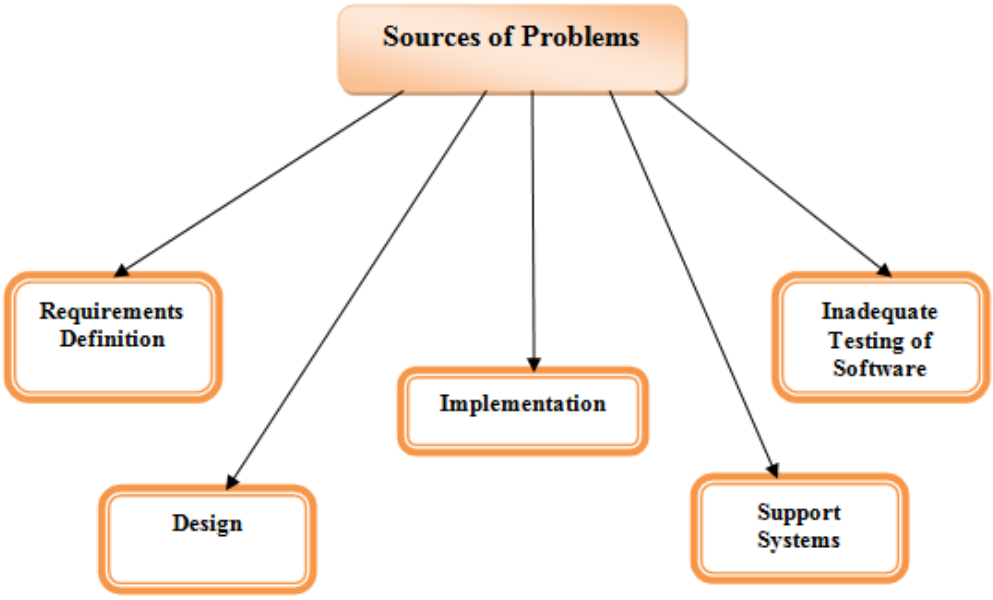

Figure 1: Various Sources of Faults

3. RELATED WORKS

3.1 CHARACTER N-GRAM BASED IR MODEL

Sangeeta Lal et al. **[1]** presented a technique for fault localization based on a Character N-Gram based Information Retrieval (IR) Model. The approach presented in this work makes use of low-level character-level representation. This work frames the problem of fault localization as a relevant document(s) search task for a given query and examines the application of character-level n-gram based textual features derived from error reports and source-code file attributes. Then, proposed IR model has been implemented and its performance is evaluated on dataset of two popular open-source projects, i.e. JBoss and Apache.

Figure 2 depicts the architecture of fault localization using Character N-Gram based IR Model. The system comprises of 3 components: (1) Character N-gram Based Feature Extractor, (2) Similarity Computation and (3) Rank Generator. The feature extractor module extracts all the character N-grams from error reports and source code files. The similarity computation module calculates the final score between fault report and source code. Lastly, rank generator is applied.

The accuracy of the proposed system is measured in terms of the commonly used SCORE and MAP (Mean Average Precision) metrics for the purpose of bug localization. It has been observed that the median value for the SCORE metric for Apache and JBoss dataset is 93.70% and 99.03% respectively. Moreover, the average precision value is between 0.9 and 1.0 for 16.16% of the bug reports in the JBoss dataset and for 10.67% of the bug reports in the Apache dataset.

3.2 NEAREST NEIGHBOR QUERIES

Manos Renieris and Steven P. Reiss **[10]** proposed a technique for bug localization based on distances and differences between abstractions of program runs. In this, a general architecture for error localization systems is defined which is based on spectra and models. The architecture consists of the following phases, as shown in Figure 3.

1. A classification phase.
2. A trace abstraction phase, which converts traces to abstract representations of runs.
3. A modeling phase, which converts the successful spectra into a model of successful runs, perhaps taking into account the failing spectrum.
4. A differencing phase.
5. A source mapping phase.

Moreover, they have defined a novel assessment strategy for error localization systems, and used it to compare four instantiations of the architecture: three based on coverage spectra (using a union, an intersection, and a nearest neighbor model), and one based on a new abstraction of profile data and the nearest neighbor model. The observations illustrated that the intersection model performs poorly whereas the union model is bimodal. The nearest neighbor model outperforms them on average, but it only becomes effective for the more elaborate spectra.

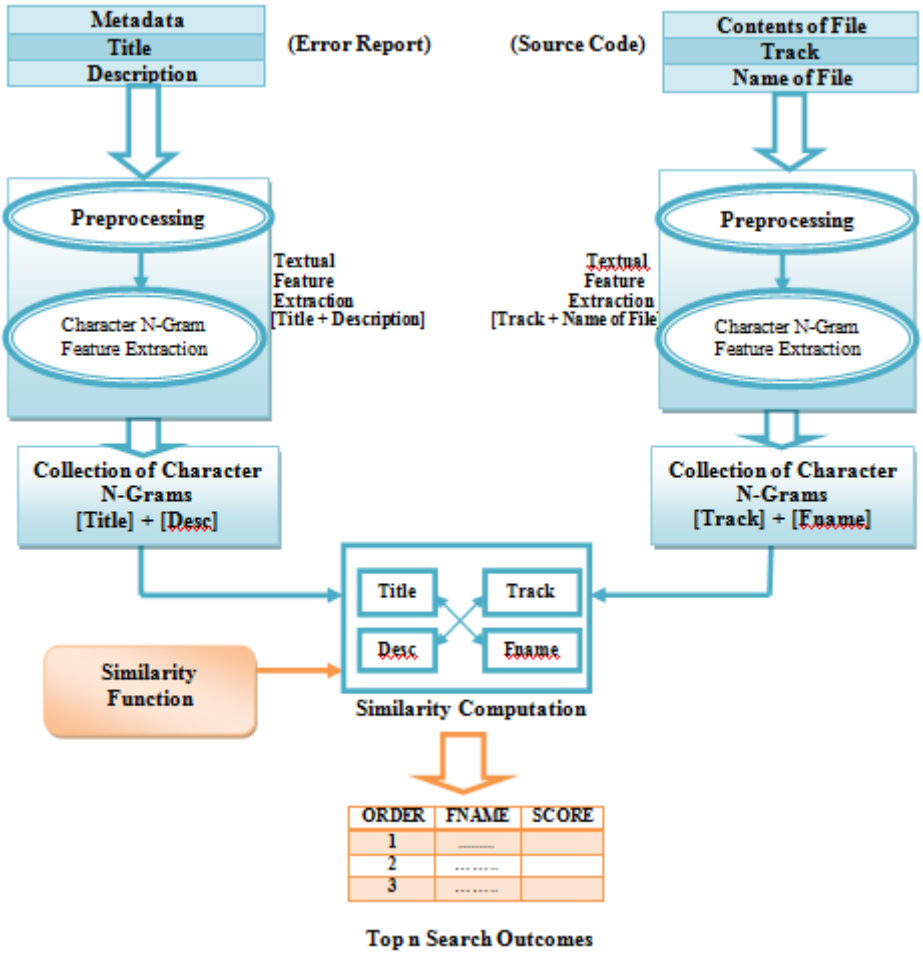

Figure 2: Fault Localization using Character N-Gram based IR Model

3.3 FORMAL CONCEPT ANALYSIS (FCA)

Peggy Cellier et al. **[6]** proposed the application of Formal Concept Analysis (FCA) for fault localization. This process combines association rules and formal concept lattices in order to move into the source code of faulty programs. This new technique has many advantages over the existing methods. This approach computes in one pass more information about execution differences than all the other approaches together. Especially, the information computed by two methods, the union model and the intersection model can be directly found in the trace lattice.

Another advantage of this technique is that the lattice structure gives a better reasoning basis than any particular distance metrics to find similar execution traces. Actually, all differences between the sets of executed lines of passed and failed executions are represented in the trace lattice. A given distance between two executions can be computed by dedicated reasoning on specific attributes. Thirdly, this approach treats several failed executions together. Moreover, it is possible to handle traces that contain other information than lines numbers in FCA framework.

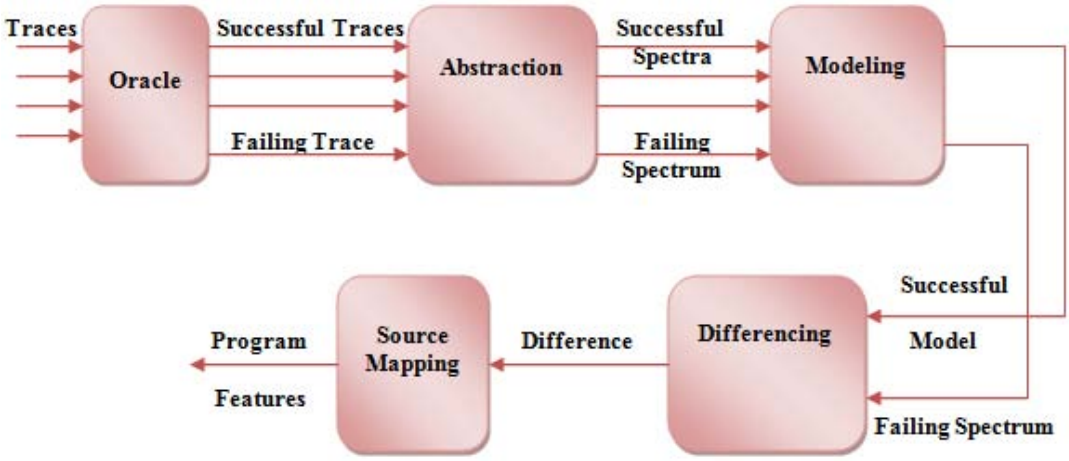

Figure 3: General Architecture for Fault Localization

3.4 PARALLEL FAULT LOCALIZATION

James A. Jones et al. **[7]** proposed a parallel debugging process that leverages the advantages of parallel work flow by reducing total developer cost and reducing the time of error localization. In this research work, two parallel debugging approaches have been presented that produce distinctive sets of test cases that help in simultaneous debugging by providing these specialized sets to different developers. Also, a novel technique is developed for organizing a useful halting criterion in the clustering of failing executions, providing an early forecasting of the number of active faults.

An analytical study has also been presented in this work that illustrates the cost saving potential of these two parallel debugging techniques as compared to sequential debugging techniques. This means that either clustering technique, which has the ability of fault localization, has potential economic benefits. This result determines the benefits of clustering executions for the purpose of localizing multiple faults. Lastly, the two metrics, i.e. total developer expense and critical expense for fault localization can provide parameters for a cost model for parallel debugging where a developer can decide the combination of factors that suits best for resources and time requirements.

4. CONCLUSIONS

Selecting a persuasive fault localization methodology generally requires expert knowledge regarding the program. Software debugging is carried out in order to find location of bugs and to achieve fault-free program. This work is an attempt to provide with the existing fault localization techniques and thus, to render help to the software tester in two aspects – i.e. cost saving and time saving. The software tester can choose any fault localization approach complying with his choice and need.

5. REFERENCES

[1] Sangeeta Lal and Ashish Sureka, "A Static Technique for Fault Localization Using Character N-Gram Based Information Retrieval Model"; Proceedings of ISEC '12, Feb. 22-25, 2012 Kanpur, UP, India.
[2] Nicholas DiGiuseppe, James A. Jones, "On the Influence of Multiple Faults on Coverage-Based Fault Localization"; ISSTA '11, July 17–21, 2011, Toronto, ON, Canada.
[3] Maneesha Srivastav, Yogesh Singh, Durg Singh Chauhan, "Faulty Slice Distribution using Complexity Estimation for Debugging in Parallel"; International Journal of Computer Applications (0975 – 8887), Volume 7– No.1, September 2010.
[4] W. Eric Wong, Vidroha Debroy, Byoungju Choi, "A family of code coverage-based heuristics for effective fault localization"; The Journal of Systems and Software 83 (2010) 188–208.
[5] W. Eric Wong, and Vidroha Debroy, "Software Fault Localization"; IEEE Reliability Society 2009 Annual Technology Report.
[6] P. Cellier, M. Ducasse, S. Ferre, and O. Ridoux, "Formal Concept Analysis Enhances Fault Localization in Software"; Proc. of the 4th International Conference on Formal Concept Analysis, pp. 273-288, Montreal, Canada, February 2008.
[7] James A. Jones, James F. Bowring, Mary Jean Harrold, "Debugging in Parallel"; ISSTA'07, July 9–12, 2007, London, England, United Kingdom.
[8] Joseph R. Ruthruff, Margaret Burnett, Gregg Rothermel, "Interactive Fault Localization Techniques in a Spreadsheet Environment"; IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 32, NO. 4, APRIL 2006.
[9] J. Ruthruff, E. Creswick, M. Burnett, C. Cook, S. Prabhakararao, M. Fisher II, M. Main, "End-User Software Visualizations for Fault Localization"; Proceedings of ACM Symposium on Software Visualization, June 11-13, 2003, San Diego, CA.
[10] Manos Renieris and Steven P. Reiss, "Fault Localization With Nearest Neighbor Queries"; Citeseerx, 2003.
[11] Rui Abreu, Peter Zoeteweij, Arjan J.C. van Gemund, "An Evaluation of Similarity Coefficients for Software Fault Localization"; TRADER project under the responsibility of the Embedded Systems Institute.