iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** IV **Month of publication:** April 2020

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue IV Apr 2020- Available at www.ijraset.com

Automatic Detection and Correction of Software Faults: A Review Paper

Prof. K. B. Vayadande[1], Dr. Nikhil D. Karande[2]
[1]*Assistant Professor, Department of Information Technology, Vishwakarma Institute of Technology, Pune, Maharashtra, India*
[2]*Associate Professor, G.H. Raisoni Institute of Engineering and Technology, Pune, Maharashtra, India*

Abstract: Faults in software document can really affect the entire behavior of the software. Many types of faults exist in the software code document after its development. Software becomes less reliable if these faults are not removed from it. A proper understanding of the different types of software faults and a classification of these faults are required before applying any techniques related to their detection or correction. Manual inspection of the code document is also possible but disadvantage of this technique is, it requires too much time and manpower. So employing large teams for inspection of code is very costly. So, Automatic detection and correction of these software faults, which suffer from lack of automated tools, are vital to ease the maintenance of software. Most of the techniques available today only focuses on the detection of fault, so in order to remove these faults in significant and reliable way is a challenging task. So many techniques are available for detection and corrections. This paper discusses these techniques.

Keywords: Fault, Inspection, Software Document, Reliable

I. INTRODUCTION

The process of implementing software without any kind of fault is challenging**.** Currently, the different compilers/interpreters for software languages have been progressively improved. The presence of the different types faults in software can increase the number of software failures and can thus decrease the reliability of software. Of course, the software reliability is enhanced if the risks of software failure are avoided.

Achieving reliable software is a goal of developers. In order to prevent such faults developers and software inspectors must verify software for all possible faults during the development stages, and also validate the software product before delivering it. Therefore, it challenges researchers to develop methods or techniques to detect or prevent the faults during development period in order to obtain a high level of reliability for software product.

Currently many software detection techniques have been proposed and implemented. One of these techniques is code inspection, first introduced by Fagan [6]. This technique can detect the software coding errors at early stage in lifecycle. Although code inspection effect is that software quality can be improved, all the existing techniques for maintaining software reliability are reliant on the ‘‘checklist’’ approach to verify the software instructions and data sets. If the software size is small and not so complicated, the checklist process can be performed manually, otherwise it can become too unwieldy. In this paper, we have discussed several techniques for fault detection and fault correction.

II. LITERATURE REVIEW

1) *Automatic detection and correction of programming faults for software applications by Prattana Deeprasertkul, Pattarasinee Bhattarakosol and Fergus O_Brien [1]:* In this paper Precompiled Fault Detection (PFD) technique is proposed to detect and correct faults before a source code is compiled. The objective of the PFD technique is to increase software reliability without increasing the programmers_ responsibilities. The concepts of ‘‘pre-compilation’’ and ‘‘pattern matching’’ are applied to PFD in order to reduce the risk of significant damage during execution period. This technique can completely eliminate the significant faults in software and thus, improves software reliability.
2) *Analysis of Software Release Problems Based on Fault Detection and Correction Processes by Haiyan Sun, Jia He, Bingfeng Xie and Ji Wu[2]:* In this paper, the modeling of fault-correction process from the viewpoint of correction time is first discussed. By proposing a new cost model, further analysis on the optimal release time determination is presented, which is also based on the model incorporating both fault detection and correction processes. The experiment results show it is more suitable to the reality. Finally, we propose a new solution to the assignment of testing resource. The approach is also illustrated with an actual data set from a software development project.

3) *Tracking Down Software Bugs Using Automatic Anomaly Detection by Sudheendra Hangal and Monica S. Lam[3*]: In this paper they have introduced DIDUCE, a practical and effective tool that aids programmers in detecting complex program errors and identifying their root causes. By instrumenting a program and observing its behavior as it runs, DIDUCE dynamically formulates hypotheses of invariants obeyed by the program. DIDUCE hypothesizes the strictest invariants at the beginning, and gradually relaxes the hypothesis asviolations are detected to allow for new behavior. The violations reported help users to catch software bugs as soon as they occur. They also give programmers new visibility into the behavior of the programs such as identifying rare corner cases in the program logic or even locating hidden errors that corrupt the program's results.
4) *AFID: An Automated Approach to Collecting Software Faults by Alex Edwards, Sean Tucker and Brian Demsky [4]:* In this paper they present a new approach for creating repositories of real software faults. We have developed a tool, the Automatic Fault Identification Tool (AFID), that implements this approach. AFID records both a fault revealing test case and a faulty version of the source code for any crashing faults that the developer discovers and a fault correcting source code change for any crashing faults that the developer corrects. The test cases are a significant contribution, because they enable new research that explores the dynamic behaviors of the software faults. AFID uses an operating system level monitoring mechanism to monitor both the compilation and execution of the application. This technique makes it straightforward for AFID to support a wide range of programming languages and compilers.
5) *Automated Debugging and Bug Fixing Solutions: A Systematic Literature Review and Classification by Hafiz Adnan Shafiq and Zaki Arshad [5]:* The scope of work is to identify all those solutions that correct software automatically or semi-automatically. Solutions for automatic correction of software do not need human intervention while semi-automatic solutions facilitate a developer in fixing a bug. We aim to gather all such solutions to fix bugs in design, i.e., code, UML design, algorithms and software architecture. Automated detection, isolation and localization of bug are not in our scope. Moreover, we are only concerned with software bugs and excluding hardware and networking domains.

III. TECHNIQUES USED

Automatic detection and correction of programming faults for software applications by Prattana Deeprasertkul, Pattarasinee Bhattarakosol and Fergus O_Brien [1] – "PFD technique performs the fault detection as a software guard. The PFD preprocesses the programs before the compilation takes place as shown in Fig.1[1] According to the functionality defined for PFD, it consists of two main modules: detection module, and correction module. In this paper [1] author formally introduce the definitions of a set of PFD faults, a fault detection function, and a fault correction function."[1]

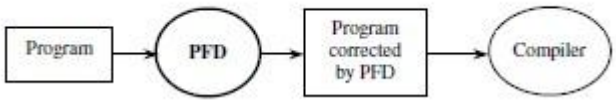


Figure 1[1]

Analysis of Software Release Problems Based on Fault Detection and Correction Processes [2] "In this paper they have proposed two main modules

A. Fault Detection Models

A fault detection process is assumed to follow a non homogeneous Poisson process (NHPP). Fault correction uses

$$m_d(t)=\int_0^t f_d(s)ds$$

…….[2]

B. Fault Correction models Based on the Correction Time

Fault correction uses

$$m_c(t)=\int_0^t f_c(s)ds\ .$$

……..…….[2]

Where - fd is Fault Detection
M_d is Mean Value Function for Detection
M_c is Mean Value Function for Correction"[2]

1) *AFID: An Automated Approach to Collecting Software Faults[4]*: "AFID automatically records software faults by monitoring the compilation and execution steps of the software development process. The underlying design principle for AFID is to record as much software fault data as possible while imposing minimal runtime overheads and requiring minimal assistance from the developer. The final goal of the AFID project is to collect fault data from a wide range of software developers working on real projects. Therefore, requiring the developer to actively participate in recording faults would potentially make finding developers to use AFID much more difficult. According to this principle, AFID has been designed to only record faults that actually cause crashes. AFID does not recognize more subtle correctness faults because that would burden the developer with describing the desired behavior of an application. We expect that we can learn much interesting information from crashing faults alone."[4]

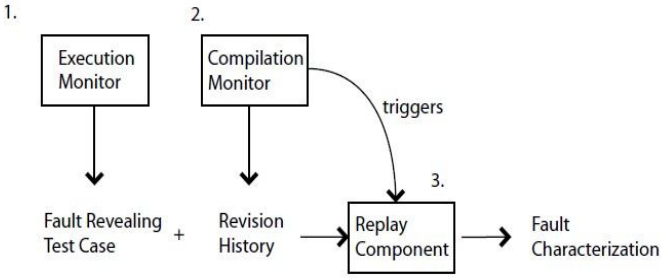


Figure. 2[4]

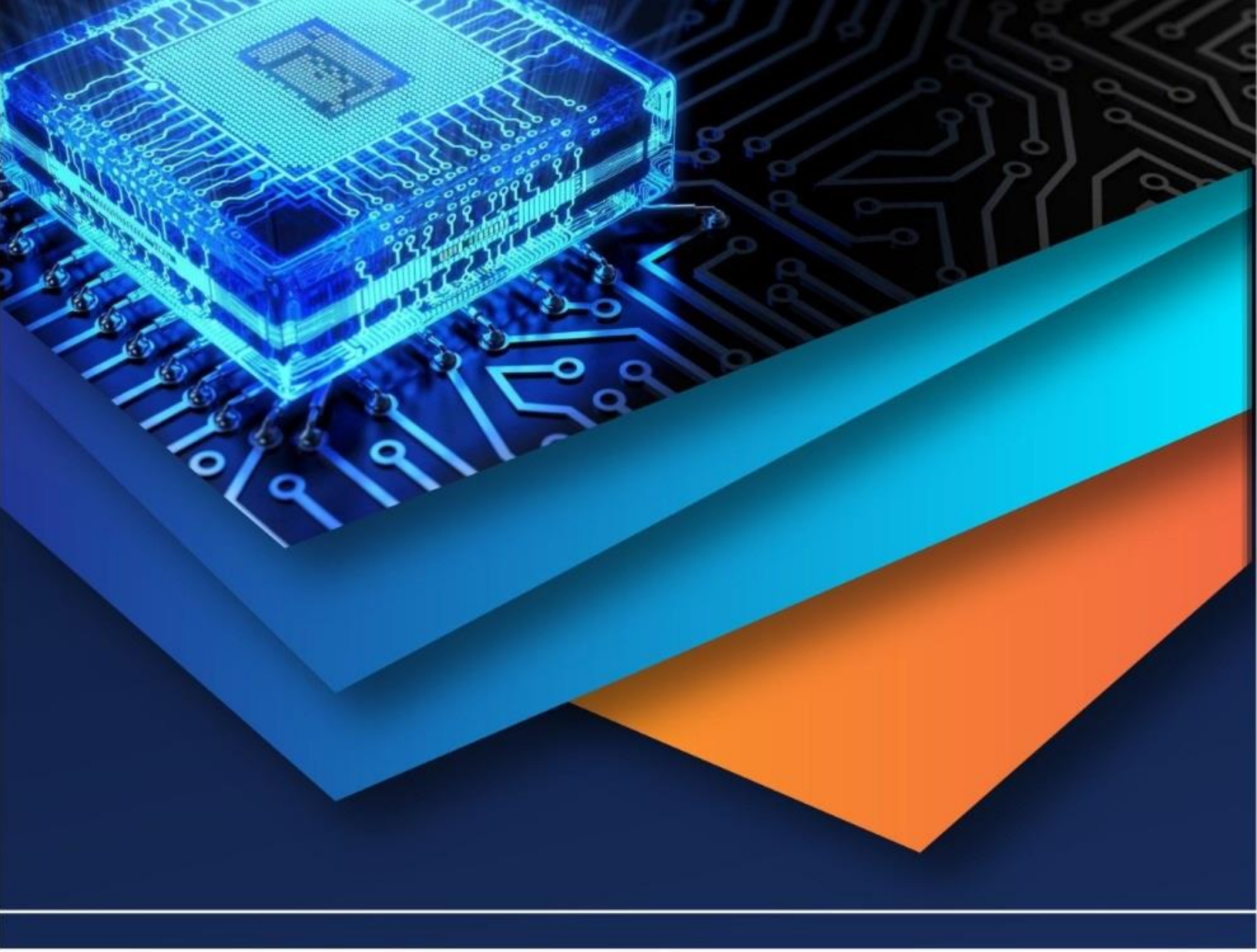
2) *Experience with Fagan's Inspection Method[6]:* "Fagan was a professional quality-control engineer who derived his basic ideas on statistical quality control from two of the gurus of quality management, Deming and Juran. The inspection process is akin to walkthroughs, although it differs significantly in some aspects. It can very briefly be described as follows. An inspection is organized by a moderator, who may be appointed by the software quality-assurance group. The moderator receives a copy of the document to be inspected and checks that it satisfies a number of predetermined criteria (entry criteria). He or she then puts together an inspection panel of no more than five people (including the author of the document to be inspected). The inspectors are then invited to attend a short (20–30 minutes) 'kick-off' meeting. At this meeting the objective of the inspection is defined, the subject matter briefly explained and any other relevant details are discussed. The inspection material is distributed and roles may be assigned to some or all members of the panel, with requests to pay specific attention to some aspect of the documents to be inspected."[6]

IV. CONCLUSION

Fault in software can affect the working of software. In some cases software might crash and in some cases it could give you wrong results. So removing faults from software becomes a vital task. In this paper survey on different techniques of fault detection and correction have been performed. Every technique works differently. These techniques have their own efficiency factor.

REFERENCES

[1] Automatic detection and correction of programming faults for software applications, Prattana Deeprasertkul, Pattarasinee Bhattarakosol and Fergus O_Brien, The Journal of Systems and Software 78 (2005) 101–110.
[2] Analysis of Software Release Problems Based on Fault Detection and Correction Processes by Haiyan Sun, Jia He, Bingfeng Xie and Ji Wu, International Conference on Computer, Communications and Information Technology (CCIT 2014).
[3] Tracking Down Software Bugs Using Automatic Anomaly Detection by Sudheendra Hangal and Monica S. Lam.
[4] AFID: An Automated Approach to Collecting Software Faults by Alex Edwards, Sean Tucker and Brian Demsky, Automated Software Engineering Journal manuscript
[5] Automated Debugging and Bug Fixing Solutions: A Systematic Literature Review and Classification by Hafiz Adnan Shafiq and Zaki Arshad, Faculty of Computing Blekinge Institute of Technology SE-371 79 Karlskrona Sweden, Thesis no: MSSE-2014-06
[6] Experience with Fagan's Inspection Method,E. P. DOOLAN Shell Research B. V., P. O. Box 60, 2280AB Rijswijk (Z-H), The Netherlands



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY