Detecting Duplicate Bug Reports and Identifying Bugs using Bug Tracking System

Sanjana K S, Mruthula N R, Rupesh Ravi M R

PG Scholar,Department of CSE,TKM Institute of Technology,Kollam,Kerala, India.
Assistant Professor, Department of CSE,TKM Institute of Technology,Kollam, Kerala, India.
Assistant Professor, Department of CSE,TKM Institute of Technology,Kollam, Kerala, India.
sanjanasivan16@gmail.com, mnrieeepjt2013@gmail.com, ,rupeshd2k@gmail.com

Abstract— Bug tracking frameworks play an vital part in program upkeep. They permit clients to yield bug reports. The proposed by S´liwerski, Zimmermann, and Zeller (SZZ) for recognizing bug presenting changes is at the establishment of a few inquire about ranges inside the computer program designing discipline. In Open Source Program improvement there is a shared understanding among open source developers, reporters and clients that effectively moves forward the item quality. Prior comes about propose that the current SZZ implementations still need instruments to precisely distinguish bug-introducing changes and copy bug discovery. Developers utilize the data given in bug reports to distinguish the cause of the imperfection. Copy bug report sections in bug trackers have a negative affect on program and a significant sum of time is misplaced in copy bug report investigation. In this way the progressed SZZ framework points to characterize mechanized copy bug reports location by utilizing a hashing calculation. The results obtained from the linked bug reports and SZZ approach are compared and analyzed. Our proposed system gives a efficient mean for assessing the information that is produced by a given a bug tracking systen execution utilizing bug reports and bug settling changes for expanding the efficiency of software support activities.

Keywords— Data mining, SZZ approach, bug fixing changes, bug inducing changes, bug tracking system , fix-inducing commits etc.

I. INTRODUCTION

Software bugs are expensive to settle. For instance, a later consider recommends that developers spend around half of their time settling bugs. Thus, lessening the required time and exertion to settle bugs is an appealing investigate issue with bounty of potential for mechanical affect. After a bug has been detailed, a key assignment is to recognize the root cause of the bug such that a group can learn from its mistakes. Consequently, analysts have created a few approaches to recognize earlier bug-introducing changes, and to utilize such information to dodge future bugs. Many program projects depend on bug reports to coordinate remedial upkeep action. In open source software projects, bug reports are frequently submitted by program clients or developers and collected in a database by one of a few bug following instruments. Permitting clients to report and possibly offer assistance settle bugs is accepted to make strides computer program quality in general. Bug tracking frameworks permit clients to report, depict, track, classify and comment on bug reports. 1) Software engineering

The reason of a logical, restrained, quantitative, approach to the event, method and support of software framework is named software designing. The software designing was popularized for the term of the NATO program designing conference in 1968.It envelops techniques and methods, regularly directed by a program advancement hone, with the reason of making strides the unwavering quality and viability of program frameworks. The specialist of computer program generation incorporates data, devices and methodologies for software framework needs, plan, development, testing, and support assignments. Software framework designing is related to the teach of computing, designing, administration, number-crunching, extend administration, program framework bioengineering, and frameworks engineering.

2) Common issues in computer code development:

a) Destitute prerequisites: In the event that necessities aren't clear, fragmented, as well wide or not testable, there'll be issues.

b) Unreasonable plan: In case an excessive amount of work is packed in inadequately time, issues are usual.

c) Insufficient testing: no one can recognize whether or not or not the program is any savvy quality till the client complaints or frameworks collapse.

d) Futurities: prerequisites to stack on unused alternatives once improvement is current, exceptionally common.

e) Miscommunications: In the event that developers don't recognize what's required or clients have wrong desires, issues are bonded.

3) Testing to progress program quality: To preserve computer program quality testing is a major stage in SDLC. Testing meets three objectives.

a) Distinguishing proof of Mistakes: These are self-evident inconsistencies that appear up inside the behavior of program or a substance or a module. Such behavior since of the taking after is taken into account error. Off-base add up to, course of action, messages that say off-base issue, activity that don't execute as guaranteed: the erase button doesn't erase, the overhaul menu doesn't upgrade properly

b) Conformance to requirements: These errors are the results of testing the functions within the software system against demand the need Definition Document to confirm that each requirement, functional or non-functional is within the system which it works properly. Typically this can be often referred to as associate Operational Qualification (OQ). However note that although a number of the requirements don't appear to be “Operational”, this is often associate operational check. For elements, if the message are not show, then there's an absence of conformity and therefore the system doesn't operate properly.

c) Performance Qualification: These aren't “errors” as such however failure to evolve to Performance Qualification

(PQ) became a regular methodology of testing for past reasons. Some systems can perform otherwise below completely different masses and condition. For instance, a subject search application may have to control among specific time reply for a load of up to three hundred queries an hour. The software perform is also designed properly ie, might pass the operational qualification, however might fail to fulfill the required masses attributable to poor programming or too several information calls.

Static analysis tools are an alluring way to discover infringement of code quality and security requirements. They can analyze a program without running it, and when they work soundly, they can find all occurrences in a lesson of defects. But they as well frequently convey off-base takes note or takes note that designers do not care nearly. Various present day devices are not by and huge sound, but basically put a few effort into finding plans of issues that have been seen a few time recently. They utilize heuristics to play down the number of wrong notices and sharpen in on notices that are likely to be curiously to engineers. Discover Bugs is one such instrument, with over 150 locators composed to discover around 400 designs of inadequate behavior. We can discover these botches by checking on source code in software configuration management (SCM) frameworks. But we may not need to look at each modification or commit; fair those that are straightforward, those that settle bugs and those that present bugs.

Bug introducing commits may be related with unit test regressions, or may be found by following lines changed in bug fixing commits to the point in time when they were already altered. These past alterations are now and then called fix-inducing commits. One issue with connecting between SCM frameworks and issue following databases is that not all issues in the databases are related with settling bugs; a few are demands for modern features while others remind developers of required errands. Indeed when a report demands a bug settle, the related source code changes may incorporate other exercises such as including test cases or refactoring code, and so cannot continuously be respected as bug-fix changes. And indeed when the source code changes are settling bugs, the fixes may not have been specifically initiated by a past adjustment of the same lines, but may instep have been caused by an API alter or actuated by a partitioned bug settle. Our survey recommends that applications and algorithms distinguishing fix-inducing commits may require to be more particular in how they select bug settling commits since now and then it is vague whether the settle could have been anticipated by more cautious programming of the changes that already altered the affected lines.

II. RELATED WORK

Defect Tracking Framework has been created for the improvement of software quality [1]. There are of different existing procedures like Redmine, Bugzilla, Mozilla etc. which doesn't meet the criteria of culminate defect tracker. The point of the paper is to make an online defect tracking framework profitable for applications made in an organization. The Defect Tracking framework is a web based system that can be gotten to all through the organization. In this system can be utilized for sorting defects against an application module, distributing defects to individuals and following the surrenders to determination. This solicitation contains features like email notices, client support, client access control, report generators etc. This paper has been arranged to be having the view of disseminated plan, with centralized storage of the database. The measures of security and data securing component have been given a tremendous choice for fitting methodology. The requesting takes care of diverse modules and their related reports, which are made as per the appropriate procedures and standards that are put sent by the authoritative staff. This framework will provide a solution for all issues in existing bug tracking framework.

The number of defect reports regularly exceeds the assets accessible to address them [2]. Develop program projects are constrained to transport with both known and unknown bugs; they need the improvement assets to deal with each defect. We propose a method to minimize bug report triage taken a toll by evacuating the copy bug reports as they are detailed. We construct a classifier for approaching bug reports that combines the surface highlights of the report and chart clustering calculations to recognize copies. It is based on the presumption that important words are recognized not as it were by the number of times they show up in a certain content, but too by the reverse of the proportion of the records in which they show up in the corpus. In our dataset, copy bug reports of the same basic defect are no more likely to share “rare” words than are otherwise-similar disconnected sets of bug reports.

The calculations are consequently and precisely recognize bug-introducing changes [3]. The wrong positives and untrue negatives are evacuated by utilizing explanation charts. After recognizing bug fixing changes, SZZ uses a diff instrument to decide what changed in the bug fixes. The diff device returns a list of locales that vary between the two records; each locale is called a “hunk”. It observes each hunk in the bug-fix and expect that the erased or altered source code is the area of the bug. Finally, SZZ approach recognizes the beginnings of the erased or altered source code in the hunks utilizing the built-in comment work of SCM (Source Code Management) frameworks. The comment computes, for each line in the source code, the most later amendment where the line was changed, and the designer who made the change. These revisions are distinguished as bug introducing changes.

The SZZ calculation [4] is utilized to distinguish bug-introducing changes. SZZ to begin with finds bug-fix changes by finding bug identifiers or significant catchphrases in modify log substance, or taking after an explicitly recorded linkage between a bug following system and a specific SCM commit. SZZ at that point runs a diff device to choose what changed in the bug-fixes. The diff device returns a list of regions that contrast in the two records; each locale is called a hunk. It recognizes each hunk in the bug settle and acknowledge that the source code eradicated or changed in each hunk is the range of a bug. At long final, SZZ tracks down the roots of the deleted or balanced source code in the hunks utilizing the built-in clarify highlight of SCM systems. The comment on include computes, for each line in the source code, the most afterward amendment in which the line was changed, and the designer who made the change. The found roots are distinguished as bug-introducing changes.

This paper presents the progressing research work [5]. The core of our study is focused on bug fixing and bug seeding process, concretely, in the study of the assumption made in the literature which says that a given bug was introduced by the lines of code that were adjusted to it. After investigating manually and in detail how a number of bugs were presented in two diverse projects, we are creating an approach to and the Bug introducing alter consequently. Moreover, based on the over assumption we are carrying out a systematic literature survey approximately the utilize and validity of this suspicion in past studies. This effectively able to distinguish changes in adaptation control that actuated of bugs making the well-known calculation SZZ. The primary suspicion of this calculation is based on the thought that adjusted or expelled lines in a fixing-commit are the ones suspicious of inducing the later. Thus, tracing back them in the source code management framework to the time when they were modified or added result in the commit that is considered as the cause of the bug.

Any program bugs can be solved by utilizing a bug tracking framework [6]. In this framework we have designed different types of client authorization like developer, tester having different rights to associate the program. The administrator, user can make the client account in the framework and gives the permission as well as he can keep up bug following in the framework for all projects. As computer program development is completed, the projects are goes in the next stage i.e. testing. The test engineers working in the extend test the report in case any bug found they log that bug with ID and set the need with the depiction. After the first cycle of bug following is completed the designer working in the projects can log in to framework and get the bug list with priority. This can solve the bug and alter the status of that bug by designer. The administrator can get the idea of bug status, work status of both developer and tester and time span for that project. There is moreover one major part of client side to test the software called as User Accepting Testing. On the off chance that the client faces any issue in the framework after conveyance they can too log in to framework and put their issue or input regarding that extend.

On time defect reporter and scheduler framework extend gives bug tracking, offer assistance work area, issue raising, search facility, helps data and issue determination [7]. Issues related to program projects can be raised, followed and resolved by representatives. The diverse groups and representatives can connected each other through this framework. The issue following framework does all the employments that are done in routine framework. In this project it is done in more formal and productive way. All the clients like programmer and analyzer and project manager of organization can connected with each other through the defect reporting tool. This framework acts as an interface between the representatives subsequently enabling them to forward their issues to the centralized report following framework. Subsequently, making the work simple for both the issue with admin and the programmer. Defect Tracking Frameworks are primarily required to join with very other data systems like contact data, client databases, project planning systems, requirement management packages. This method is greatly handled to implement with security measures like granting admittance only to particular groups and permitting only authorized persons to modify the process.

Upkeep of the available framework is an vital handle in the Software Improvement Life Cycle (SDLC) [8].Keeping track of the problems rise in the existing and who is assigned with the maintenance work. The solutions provided for the problems are always a tedious job. The Bug Tracking and Reporting System are developed to handle these tasks and maintaining software quality. It is a Multi User online system span over intranet/internet. It is involved in almost all the stages of SDLC such as Requirement Analysis, Design, Coding, Testing and Maintenance. Based on this process the system will maintain a database for each project, in that it maintain project details, employees working on the project, and the bugs in the project which are reported by Testers. While the Tester is in the process of finding the new bugs, Bug tracking developers read the reported bugs and fixes them and updates the same information in the bug reports. Thus, the system is primarily designed to report and track bugs simultaneously to obtain software quality and deliver reliable software. This system helps organization to maintain errors, bugs and defects occur at SDLC, which helps in upcoming projects.

Expectation on a fine-grained level [9] and it speaks to the methodology level is required for various curiously comes about compared to coarse-grained expectation. These results incorporate great execution when considering quality assurance efforts, and modern discoveries approximately the relationships between bugs and histories. The major challenge for fine grained prediction procedure is to get histories from existing version control framework. We have created a fine-grained form control framework for Java, Historage to maintain a strategic distance from this issue. With this framework, we target Java computer program and conduct fine-grained forecast with well known verifiable measurements. The results show that fine-grained expectation beats coarse-grained forecast when taking the efforts essential to discover bugs into account. This appear that the past bug information does not contribute to method-level bug expectation by utilizing a relationship investigation.

The principal commitment of this paper [10] is to propose a system not as it were for extricating, but moreover to thus syncing alter logs and bugs of issue data supporting different bug tracking systems and Version Control Framework. Information and tracking of defects can be seriously deficient in nearly each open source extend, resulting in a decreased traceability of defects into the advancement logs. To plan a system that automates the handle of synchronizing and filling the gaps of the advancement logs and bug issue information for open source program projects. In specific, bug tracking information can be utilized to plan models for predicting computer program deficiencies and program reliability; issues and reliability of a computer program artefact can moreover be connected to who, when and how changes were made to it.

The inquire about consolidate how the developers experience impacts the code quality, but they disregard work burden, in show disdain toward of the reality that experienced developers are more likely to work on the more complex parts of a wander [11]. To see at work inconvenience, it can center on changed records. Utilizing thing estimations, study record complexity in each sort of record beginning. Especially, analyze three expansive commercial projects executed by the same organization to analyze the relationship between past extend involvement and developer's work. In spite of the truth that experienced creators do not persistently work on more complicated records, they show less surrenders, especially in the occasion that the differentiate in work inconvenience is not noteworthy.

Assessment is a profoundly successful but expensive procedure for quality control [12]. Most companies do not have the assets to examine all the code; hence precise imperfection prediction can offer assistance center accessible review assets. Bug Cache is a basic, award-winning prediction conspire that “caches” records that are likely to contain abandons. This paper survey the utility of Bug Cache as a gadget for centering evaluation, and at the doubts essential Bug Cache with the point of moving forward it, and at final compare it with a fundamental, standard bug-prediction procedure. At that point find that Bug Cache is in truth, important for centering audit effort; but shockingly, and find that its execution, when utilized for audits, is not much way better than a unsophisticated desire appear a illustrate that orders records in the system by their check of closed bugs and chooses adequate records to capture the lines in the system.

This paper [13] examined that bug following frameworks are vital instruments that direct the upkeep activities of program developers. The utility of this device is influenced by an over the top number of copy bug reports–in a few projects as numerous as a quarter of all reports are copies. If developers manually identify copy bug reports, this identification prepare is time-consuming and compounds the already high fetched of software maintenance. So they propose a framework that consequently classifies copy bug reports as they arrive to spare developer time.

III. PROPOSED SYSTEM

1) System Architecture

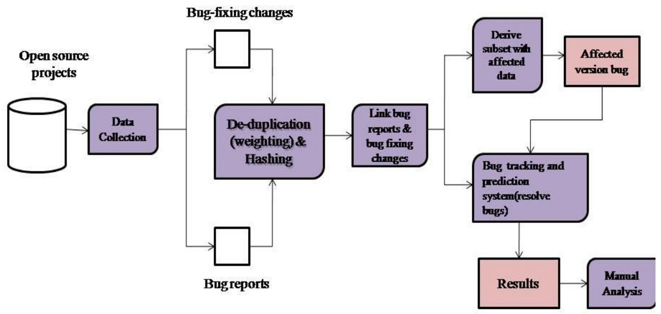

Bug-fixing changes
Open source projects
Data Collection
De-duplication (weighting) & Hashing
Link bug reports & bug fixing changes
Derive subset with affected data
Affected version bug
Bug tracking and prediction system(resolve bugs)
Results
Manual Analysis
Bug reports

Fig 1. System Architecture for filtering duplicate files.

This figure 1 provides an outline of the steps that are included in our study. At first, we collect crude information from three open source projects. This crude information is classified on the premise of predefined classes. It incorporates the bug reports and the source code changes. Next, it discover the copy records. In this way, it eliminates repetitive duplicates of information and decreases storage overhead bug reports to the bug fixing changes based on the MD5 calculation. At that point, it connect the bug reports to the bug settling changes. These approaches parse potential bug IDs inside change logs and confirm whether such bug IDs truly exist. Then infer the subset with an influenced form of information. And then pass it to the SZZ approach for execution and results are analyzed. Also at that point, the results are analyzed compared with bugs with influenced adaptation and connected bug reports. SZZ analyzes the line of code that changed to settle the bug and traces back through the code histories to discover when the changed code was presented. The results are analyzed after correcting the bugs.

2) Weighting for duplicate bug reports

It is a framework that in this way classifies copy bug reports as they arrive to save designer time. It is based on the suspicion that fundamental words are recognized not as it were by the number of times they show up in a certain substance, but as well by the speak of the degree of the records in which they show up up in the corpus.

For outline, a word like “the” may appear up various times in a single record, but will not be escalation weighted in the event that it as well appears up in most of the files. The well-known TF/IDF weighting consolidates both normal term recurrence inside a single file as well as converse archive recurrence over a entirety corpus. Thus this dataset, copy bug reports of the same fundamental imperfection are no more likely to share “rare” words than are otherwise-similar irrelevant sets of bug reports. In this way do not consolidate a weighting figure comparing to reverse report recurrence. Consequently weighting condition for literary likeness is:

Each position t in the representative vector of a bug report v is decided based upon the recurrence of term t and the consistent scaling variables present in the equation. After getting the linked bug information, we select the subset of linked bugs that have the affected form field filled. At that point execute the different SZZ usage and get the SZZ-generated information and compute results utilizing the SZZ generated information. At last, perform a manual examination based on the obtained results. To begin with, we analyze things obtained from both affected version and SZZ generated information. The objective of our manual examination is to investigate if the proposed measurements help recognize subsets of the SZZ-generated information that is suspicious.

3) Working Principles and Techniques

MD5 Algorithm

MD5 calculation can be utilized as a cryptographic hash work. Too utilized as a checksum to confirm information astuteness. Takes as input of subjective length and produces as a 128 bit “digest message” of the input. The MD5 calculation is outlining for progressed signature applications, where a colossal record maps a huge record of bits down to sensible a few bits with a private key underneath a cryptosystem to evade collisions.

```
//Note: All factors are represented utilizing unsigned 32 bits and w
//Define r as the following
```

```
 Var int[64] r, k
r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}
//Use parallel numbers portion of the sines of integers as constants:

Initialize chaining variables or Pre-processing:
for i from 0 to 63
   k[i] := floor(abs(sin(i + 1)) × 2^32)
//Initializing the chaining  variables:
Var int h0 := 0x67452301
Var int h1 := 0xefcdab89
Var int h2 := 0x98badcfe
Var int h3 := 0x10325476 //Pre-processing or appending padding bits to the message length:
append "1" bit to message
append "0" bits until message length in bits ≡ 448 (mod 512)
append bit length of message as64-bit little-endian integer to message
//Processing the message 512-bit chunks:

for each 512-bit of message
 break chunk into sixteen 32-bit little-endian words w(i), 0 ≤ i ≤ 15
//Initializing hash value:

Var int a := h0
Var int b := h1
Var int c := h2
Var int d := h3
for i from 0 to 63
if 0 ≤ i ≤ 15 then
         f := (b and c) or ((not b) and d)
         g := i
else if 16 ≤ i ≤ 31
         f := (d and b) or ((not d) and c)
         g :=  (5×i + 1) mod 16
else if 32 ≤ i ≤ 47
         f := b xor c xor d
         g := (3×i + 5) mod 16
else if 48 ≤ i ≤ 63
         f := c xor (b or (not d))
         g := (7×i) mod 16
      temp := d
      d := c
      c := b
      b := ((a + f + k(i) + w(g)) left rotate r(i)) + b
      a := temp
//Add this chunk's value hash to the result:
   h0 := h0 + a
   h1 := h1 + b
   h2 := h2 + c
   h3 := h3 + d
var int digest := h0 append h1 append h2 append h3
```

Pseudo code for MD 5 algorithm

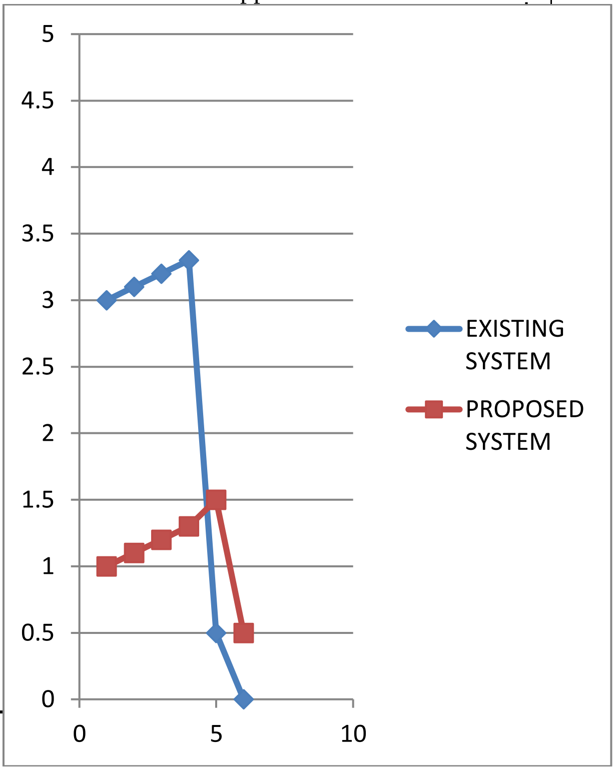
3) Bug tracking and prediction

- Identify common bugs occur in each project.
- Identify the problem when bug occurs.
- Identify the solution for each bug.
- Declare a tag number for distinct bugs.
  - a hybrid feature extraction and transformation method is proposed, which can handle various data formats existing in software history repositories.
- Description and solution for each and every bugs in tag.
- Identify common bug and put in same tag and when bug occurs it will check in tag.

IV. PERFORMANCE EVALUATION

1) Experimentation & Result Analysis

In this paper, the inputs are outlined is such a way that event of blunders are minimized to its greatest since as it were authorized client or administrator can able to get to this apparatus. The input is given by the administrators are confirmed at the section frame itself. So there is no chance of unauthorized getting to of the device. Any anomaly found in the inputs are checked and dealt with successfully. Input plan highlights are guaranteeing framework unwavering quality and create comes about from precise information. And they can result in the generation of wrong data. The results appear that our approach performs way better than the existing work by recognizing the duplicate bug reports. The success and failure of the framework depends on the yield, in spite of the fact that a framework looks appealing and client inviting, the yield it produces chooses upon the utilization of the framework. The yields created by the framework are checked for its consistency, and yield is given straight forward so that client can handle them with ease. For many end user, outputs is the main reason for developing the system and the basis on which they will evaluate the usefulness of the application.



V. CONCLUSION

In this paper, we propose a framework to evaluate the implementations of the SZZ approach. And software concern to detect and manage the bug in their products effectivelyefficiently. Utilizing bug tracking software can assist in troubleshooting errors for testing and for development processes. With the ability to provide comprehensive reports, documentation, searching capabilities, analyzing bug, tracking bugs and issues is a great tool for those software development needs. The main objective of the proposed system is to full analyze the bugs and report the same to the administrator in an efficient manner so that he can get right information at right times. The paper objective is to fully systemize everything so that the possibilities of bugs should be reduced at all levels. We propose a system that automatically classifies duplicate bug reports as they arrive to save developer time. Moreover compare the results from both the linked bug reports and SZZ approach actualized information and at that point analyze the results. The objective of our manual examination is to explore on the off chance that the proposed measurements offer assistance distinguish subsets of the SZZ-generated information that are suspicious.

Our framework is able to decrease advancement fetched by sifting out the copy bug reports. At last, our proposed system can be utilized to direct non-experts in their investigation of expansive sets of SZZ-generated information.

ACKNOWLEDGMENT

Special thanks to one and all, who helped to complete this work by giving guidance and valuable suggestions.

REFERENCES

[1] SujataSolanke, Prof. Prakash N. Kalavadekar “Defect Tracking System” International Journal of Electrical, Electronics ISSN No. (Online): 2277-2626 and Computer Engineering **3**(1): 212- 217(2014).

[2] NicholasJalbert, Westley Weimer “Automated Duplicate Detection for Bug Tracking Systems ”Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on24-27 June 2008

[3] S. Shivaji, E. Whitehead Jr, R. Akella, and S. Kim.“Reducing Features to Improve Bug Prediction”, Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on 16-20 Nov. 2009

[4] S. Kim, T. Zimmermann, K. Pan, and E. J. Whitehead, “Automatic identification of bug-introducing changes” in Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering, 2006, pp. 81–90.

[5] Gema Rodriguez-Perez, Gregorio Robles , Jesus M.Gonzalez-Barahona “New Approach To Understand The Origins Of a Bug ”Universidad Rey Juan Carlos

[6] Gauri M. Puranik “Design of a Bug Tracking System” Vol. 3, Issue 7, July 2014.

[7] K.Lavanya1, P.Jennifer “Bug Tracking System Analysis” Vol. 3, Issue 7, July 2015.

[8] K Poornachandra, G Naga Lakshmi “Bug Tracking System BTRS for Software Quality Assurance and Maintenance, Vol. 3, Issue 9, September 2015

[9] “Bug Prediction Based on Fine Grained Module Histories” Hideaki Hata, Osamu Mizuno and Tohru Kikuno Osaka University

[10] ”Towards an Automation of the Traceability of Bugs from Development Logs –A Study based on Open Source Software” Bilyaminu Auwal Romo Andrea Capiluppi Department of Computer Science Brunel University London, United Kingdom

[11] “Evaluating the Work of Experienced and Inexperienced DevelopersConsidering Work Difficulty in Software Development” Taketo Tsunoda, Hironori Washizaki, Yosiaki Fukazawa Department of Computer Science and Engineering Waseda UniversityTokyo, Japan.

[12] FoyzurRahman Daryl Posnett Abram Hindle Earl Barr Premkumar Devanbu “BugCachefor Inspections Hit or Miss?”Department of Computer ScienceUniversity of California Davis, Davis, CA., USA

[13] Nicholas Jalbert, Westley Weimer “Automated Duplicate Detection for Bug Tracking Systems” International Conference on Dependable Systems & Networks: Anchorage, Alaska, IEEE, 2008.