Automated Bug Logging For Object Oriented Methodology to Enhance Quality of Software

Ms.Itti Hooda, Dr.Prof Rajender Singh Chhillar
Department of Computer Science and Applications, Maharishi Dayanand University, Rohtak, Haryana, India
Ittihooda01@gmail.com

***Abstract*— Inspections can be used to identify defects in software artifacts. The various testing methodologies are used to enhance the quality of the software product in the case when it is used in the development phase of the software. As the defects in the design of the software directly affect the quality of the software hence it is more crucial because of these aspects (problems of correctness and completeness with respect to the requirements, internal consistency, or other quality attributes). Using various reading techniques available allows the developer to go through the defects at very early stage of the development in the case of the object-oriented development.

Each reading technique in the family focuses the reviewer on some aspect of the design, with the goal that an inspection team applying the entire family should achieve a high degree of coverage of the design defects. In the field of software development, the bug tracking is being considered as the essential and most important part. The bug tracking system provides some information for the identified bugs for which the developer can rectify them or removes them. The bulk of information provided in the bug reports may cause problem to developers in ascertaining poorly designed information.

Keywords- **Bug tracking, reading techniques, Testing Methodologies, Software Testing Life Cycle, Testing Frameworks, Automation Testing, Test optimization, Quality Metrics.**

I. INTRODUCTION

Software inspections have been shown to be a practical method of ensuring that software artifacts, created during the software lifecycle, possess the required quality characteristics. For instance, inspections have been used to improve design and code quality by increasing defect removal during development. Along these lines, investigations help diminishes deserts in a product framework by guaranteeing that the product ancient rarities which are vital for its development accurately mirror the requirements of partners. Programming investigations plan to ensure that engineers manage finish, predictable, unambiguous, and remedy antiquities.

Albeit most usually connected to code records, programming examinations have additionally been utilized at before phases of the product lifecycle (e.g. in the prerequisites stage) to recognize potential issues as ahead of schedule as could reasonably be expected. Most distributions concerning programming investigations have focused on the best number and association of assessment gatherings while accepting that individual analysts can successfully distinguish deserts in programming reports without anyone else. Nonetheless, there has been experimental confirmation that group gatherings don't add to finding a critical number of new imperfections that were not officially found by singular analysts [1].

"Software reading techniques" endeavor to build the adequacy of individual analysts by giving rules that can be utilized to look at (or "read") a given programming ancient rarity and distinguish surrenders. There is exact confirmation that product perusing is a promising procedure for expanding programming quality for various circumstances and records sorts, not simply restricted to source code. It can be performed on all reports related with the product procedure, and is a particularly helpful technique for recognizing deserts since it can be connected when the records are composed.

As a rule, abandons in programming ancient rarities can be named discarded, vague, conflicting, off base, or unessential data. Be that as it may, such a rundown and the meanings of each of its things clearly should be custom fitted to the particular curio being investigated. [2]

No product is impeccable, it implies that product may require extra module or upgrade of the current module or a portion of the module may contain some unnoticed or unchecked bug that are left in the product every now and then. The bug may get presented in any period of the product improvement, i.e. necessity investigation (RA), plan (SD), coding (SC), testing (ST), usage (SI) and support of the framework (SM).

With the expansion of the engineer in open source ventures who are consistently contributing toward the undertaking advancement and change, there is a plausibility of presentation of new bugs in the task on account of the fuse of the present accommodation. There are various bugs day by day submitted to the venture site that might be utilizing some product design administration (SCM) instruments for the rendition and discharge administration of the product. The SCM instruments may not give any thought regarding the bug announcing and additionally advance of settling of bugs. There is a dire need to plan and execute the best bug/issue following and revealing framework. [3]

The bug revealing/following framework ought to give an intelligent online stage for bug detailing and following the advance. The framework may include a bland procedure or particular timetable of bug detailing process.

The generic process of the reporting a bug may generally contain the following information:

- **Title:** Title of the bug.
- **Description**: Detailed description of the bug including what, where, why, how and when the bug occurs. Actual message that appears during the operation may be included with the actual set of input and expected output.
- **Version**: Specify the version of the project.
- **Component:** Specific component need to be specified.
- **Screenshot/Attachment:** Corresponding screenshot can also be uploaded as a .jpg or .gif file by capturing the actual operation/output/message.
- **Priority**: Priority may be assigned for its urgency.
- **Severity:** Specify frequent occurrence and its impact in the system.
- **Status:** Current status of the bug (new, opened, confirmed, closed, etc.).
- **Created by:** Name of the person or id already registered with the system who is reporting the bug.
- **Assigned to:** The bug reporter may also assign bug to specific person, if one knows about a particular person who can solve this problem otherwise assigned by the moderator.
- **Revision History:** If the bug is earlier reported, show the historical changes.
- **Estimated time:** The estimated time may be specified, generally used in case of closed team environment not in the open source environment.
- **Comments:** Any other information that will be helpful in identifying the bug. [3]

II. RELATED WORK

Till date, there are very few studies on finding such relationship taking the severity of faults into account. The most popular study in the field of fault severity has been done by the authors. They examined the execution of models at three levels of seriousness, for example, high, medium and low seriousness blames and inferred that the model anticipated at high seriousness deficiencies is less precise when contrasted with the models anticipated at different severities. It was outlined that the models anticipated utilizing DT and ANN strategies are superior to anything the models anticipated utilizing LR strategy. Additionally, it was seen that the measurements, for example, CBO, WMC, RFC and SLOC are essential for the issues at all the seriousness levels.

Bolster Vector Machine (SVM), course connection organize, bunch technique for information taking care of polynomial strategy, quality articulation programming) [4]. AUC figured the execution from a ROC Curve. Numerous analysts utilize ROC bends for the modules for anticipating shortcoming inclined or not blame inclined.

In this examination, [5] the creators naturally allot needs to Firefox crash reports in Mozilla Socorro server in view of the recurrence and entropy of the accidents. At the point when Firefox fizzles, crash reports are consequently submitted to the Socorro server, and it contains a stack follow and pertinent data about nature to enable designers to investigate the crash. In the examination, the creators explored bug reports that are physically put together by clients.

These reports are unique in relation to a crash report, a bug report contains regular dialect portrayals of a bug and won't exclude any stack follow or condition data. They utilize a content mining based answer for dole out needs to bug reports. A few investigations [6] bunch bugs into various classes. Huang et al. propose a content mining answer for ordering bug reports as ability, security, execution, dependability, prerequisite, or ease of use related bugs.

In the year 2012, [7] Tian additionally anticipated the seriousness of bug reports by utilizing the closest neighbor way to deal with foresee fine-grained bug report marks. Not the same as work by Menzies which examinations an accumulation of bug reports in NASA, Tian applies the answer for a bigger gathering of bug reports comprising of more than 65,000 Bugzilla reports. Seriousness levels are given by clients, while need levels are relegated by designers.

Seriousness levels identify with the effect of the bug on the product framework as comprehended by clients and need levels identify with the significance bug reports that are gotten. Tian [7] utilized naturally identifies and dissect bug reports, which had been accounted for in the previous seven days with seriousness levels, and taking these reports in thought seriousness levels to recently announced bug reports are appointed. They decide the similitude between utilized copy bug reports with relative data and components. This comparability in data helps in allotting the seriousness levels precisely and rapidly.

In the region of comparative bugs, an examination is displayed by [8]. Some exploration in the field of bug triaging support by prescribing fitting engineers to settle a specific bug. Levenberg-Marquardt (LM) calculation in view of neural system predicts the Software defects at a beginning period of the SDLC are portrayed. They utilize information accessible on the PROMISE store of observational programming designing dataset. The dataset utilizes the CK (Chidamber and Kemerer) measurements.

The investigation reasons that the Levenberg-Marquardt (LM) neural system based calculation gives better precision (88.09%) when contrasted with each polynomial capacity based neural system (pF-NNs), straight capacity based neural system (lf-NN) and quadratic capacity based neural system (qf-NN) individually. A deformity expectation demonstrates utilizing information mining. Gayathri [9] dealt with an upgraded Multilayer Perceptron, Neural Network strategy are proposed, in which near examination of demonstrating of deformity inclination expectations utilizing a dataset of various measurements from NASA MDP (Metrics Data Program) was performed.

Different assignments of Tian [7] gave a mechanized approach the assistance of machine learning in recommending a need level by data in bug reports. They consider multifaceted fleeting, literary, creator, related reports, item and seriousness, as potential variables, which influence the need level of revealed bug reports. They utilize these components as elements to prepare a model with the assistance of an arrangement calculation (thresholding and direct relapse), which can perform well in ordinal class marks and imbalanced information. They led their investigation on more than 100,000 reports gathered from Eclipse. This test demonstrates a change of 58.61% with respect to normal F-measure by beating benchmark approach. [10] Extended their past work by utilizing separated elements to prepare a discriminative model by means of another characterization calculation (direct relapse) and their structure named DRONE. The new work gives an approach to deal with ordinal class names and imbalanced information. They figured out how to enhance their function on 100,000 bug reports from Eclipse in regards to F-quantify by 209%, which beat gauge approach.

Likewise, [11] exhibited an approach, in which they actualize an enhanced strategy for recognizing copy bug reports utilizing the literary similitude elements and twofold grouping. They utilized an aggregate of 25 literary elements, at that point run their order strategy to classify sets of bugs into two sorts: copy and non-copy. They utilized new literary components, inferred in light of content comparability measures, and prepared a few twofold order models. In the wake of preparing the models, they tried their work on bug reports gathered from Eclipse, Open Office and Mozilla to investigate the viability of the enhanced strategy. They likewise contrasted it and the present best in class and featured the similitudes and contrasts. They could accomplish a change of 6.32% in copy bug report location even without considering setting based elements.

III. DEFINING THE PROBLEM

The problem in the older system can be defined as the whole project maintenance, user's maintenance and their assignment has to be maintained manually. The Software development organizations need to confront a considerable measure of issues while keeping up physically all the support of the tasks, their bugs and their status. This sort of issue makes the entire framework a wasteful one and in this manner making a poor and sloppy working. Keeping in mind the end goal to evacuate this kind of issue, So that the paper is intended to create. Bug following programming is a "Defects Tracking System" or an arrangement of contents which keep up a database of issue reports. Bug following programming enables people or gatherings of engineers to monitor extraordinary bugs in the item adequately. Bug following programming can track bugs and changes, speak with individuals, submit and audit fixes, and oversee quality confirmation. This electronic business application is an incredible instrument for relegating and following issues and errands amid programming advancement and whatever other activities that include groups of at least two individuals.

IV. BASIC MODELS OF OO DESIGN

Requirements documents generally consist of a textual description of the functional and non-functional requirements for the software product and the related scenario definitions (i.e. use-cases). The objective of very much portrayed prerequisites is to speak to the issue to be understood by the product framework, not to determine a specific arrangement. Actually, a specific arrangement of prerequisites could possibly bolster different answers for the issue.

Conversely, programming outline exercises are worried about the portrayal of certifiable ideas that will be a piece without bounds computational arrangement of the issue. (This is genuine paying little mind to the product lifecycle demonstrate utilized.) High-level outline exercises manage the issue depiction however don't consider the limitations with respect to it. That is, these exercises are worried about taking the utilitarian prerequisites and mapping them to another documentation or shape, utilizing the worldview develops to speak to the framework through plan charts rather than only a printed portrayal. Such an approach enables engineers to comprehend the issue as opposed to endeavor to explain it. Low-level plan exercises manage the conceivable answers for the issue; they rely upon the outcomes from the abnormal state exercises and nonfunctional prerequisites, and they fill in as a model for the code. Our advantage is to characterize perusing methods that could be connected on abnormal state configuration reports. We feel that audits of abnormal state plans might be particularly profitable since they help to guarantee that engineers have satisfactorily comprehended the issue before characterizing the arrangement. (That is, our accentuation is on abnormal state understanding as opposed to low-level points of interest of the engineering.) Since low-level plans utilize a similar essential outline set as the abnormal state configuration, yet utilizing more detail, surveys of this kind can help guarantee that low-level plan begins from an astounding base.

To help the development of the plan outlines we picked UML [12] as the fundamental documentation. UML is viewed as a notational approach and does not characterize how to compose improvement errands. In this way, it can be custom fitted to various improvement circumstances. We needed to concentrate our perusing strategies on the accompanying abnormal state configuration outlines: class, cooperation (succession and joint effort), state machine and bundle. More often than not, these are the primary UML charts that engineers work for abnormal state OO outline.

They catch the static and dynamic perspectives of the issue, and even enable the cooperation to be composed, in view of bundling data. A class portrayal format was characterized to enable designers to record every one of the definitions that they used to manufacture the models. This portrayal is fundamentally the information word reference of the entire outline and can be utilized to help the perusing exercises. As abnormal state exercises don't manage the low-level issues with respect to the arrangement, organization and exercises outlines were not utilized as of now. The photo does not endeavor to demonstrate a successive procedure see but rather outlines the interdependencies among the plan exercises; bolts indicate how a few charts are utilized as contribution to assemble new ones. [2]

V. OBJECTIVES

The broad objective of the study is to present a comparison of six, different Bug Tracking System. The specific objectives of the study are:

(i) To perform the comparative study of Bug Tracking system.

(ii) To identify the limitations of Bug Tracking System under study.

(iii) To propose a framework for Bug Tracking tool.

VI. NEED AND SCOPE OF STUDY

With the increase in the use of open source software the information technology era has given birth to new revolution. Wide range of Open Source Software is available in any usage area. Wide range of software products namely Operating System, Webservers, Word Processor, Databases, Bug Tracking System, Antivirus, Data Mining Software etc. are available. The different Bug Tracking Tool are accessible in Open source space. Bug Tracking framework gives the total data about the bugs which help the designer to monitor bugs in the product item. The product item now a day are winding up more mind boggling and it is ending up more hard to monitor tremendous measure of bugs in programming having the entire and precise data about the bugs encourages the designers to determine it. So picking a decent following framework for any of item will build the efficiency of programming, enhance the correspondence between the designers, create the solid and secure programming and it will raise the consumer loyalty.

VI. PROPOSED METHODOLOGY

The normal bug detailing rates are very high in huge open source programming ventures. For instance, in the overshadowing venture, normal bug detailing rate is 50 issues for every day. In this manner, it is extremely troublesome for the mediator to investigate every last bug and after that refresh its status and task of the bug to a specific individual. The bug detection in the proposed methodology is being done using pattern matching, following are type of bug patterns in the software module:

- Cloneable Not Implemented Correctly,
- Dropped Exception,
- Suspicious Equals Comparison,
- Bad Covariant Definition of Equals,
- Equal Objects Must Have Equal Hashcodes,
- Null Pointer Dereference, Redundant Comparison to Null.

Considering an example, the variable *x* is checked against null while it is unconditionally dereferenced afterwards. From this snippet of code, we could infer that the programmer's awareness of the variable *x* could be null and this piece of code should return 0 if it indeed happen.

We collect these mistakes and create a bug pattern – "a pointer dereferenced after it is explicitly checked against null and under the true branch of that check." for this mistake. Despite the fact that each framework/apparatus has its own life cycle of the bug, here a nonspecific bug life cycle can be portrayed as takes after. At the point when the bug is first detailed, its status must be set to "Unverified" as on account of bugzilla. After its initially audit, the status can be refreshed to "New" or "Allocated" in view of the past bug history databases.

Once the trustee has adjusted the bug, he or she can refresh its status as "Settled" by indicating how it was settled i.e. "settled", "invalid (not a bug)", "copy", "won't settle", and the (in) renowned "works for me". A settled bug, won't be "shut" until another person will affirm it out or it will be affirmed by the arbitrator. When it is affirmed, the bug status progresses toward becoming "confirmed" generally the status must be set to "Revive".

It stays in "Checked" state until the point that it is joined in the discharge form of the item and its status will be refreshed to "Shut". The shut bug can be revived with its status as "New" or "Unsubstantiated". There may be many bugs that are presently being settled and some of bugs that are sitting tight for determination set to be "shut". A status report can be created now and again with number of bugs pending in every classification as a status and messaged to the trustee and also submitter of the bug. So the mediator or trustee can see the workload on individual and may delay the bugs for quite a while or reassign it to others for its determination. Along these lines, the bug can be successfully settled in a powerful way.

A neural system comprises of a pool of straightforward handling units which impart by sending signs to each other over countless associations. Every unit plays out a moderately basic employment: get contribution from neighbors or outer sources and utilize this to process a yield flag which is engendered to different units. Aside from preparing, a moment assignment is the change of the weights. The framework is characteristically parallel as in numerous units can do their calculations in the meantime. Inside neural frameworks, it is valuable to recognize three sorts of units: input units which get information from outside the neural system, yield units which send information out of neural system, and concealed units whose data sources and yield signals stay inside the neural system. [13]

In PC innovation, a parser is a program, more often than not some portion of a compiler, that gets contribution to the type of consecutive source program guidelines, intuitive online summons, markup labels, or some other characterized interface and splits them up into parts (for instance, the things (objects), verbs (techniques), and their qualities or alternatives) that would then be able to be overseen by other programming (for instance, different segments in a compiler). A parser may likewise verify that the sum total of what input has been given that is fundamental. Parsing (US:/ˈpɑːrsɪŋ/; UK:/ˈpɑːrzɪŋ/), sentence structure examination or syntactic investigation is the way toward dissecting a series of images, either in regular dialect or in coding languages, complying with the standards of a formal syntax. The term parsing originates from Latin standards (addresses), which mean part (of discourse).

Inside computational phonetics the term is utilized to allude to the formal examination by a PC of a sentence or other series of words into its constituents, bringing about a parse tree demonstrating their syntactic connection to each other, which may likewise contain semantic and other data.

The term is likewise utilized as a part of psycholinguistics while portraying dialect appreciation. In this unique circumstance, parsing alludes to the way that individuals dissect a sentence or expression (in talked dialect or content) "as far as syntactic constituents, recognizing the parts of discourse, syntactic relations, and so forth." This term is particularly normal while examining what phonetic prompts help speakers to decipher plant way sentences.

Inside software engineering, the term is utilized as a part of the examination of coding languages, alluding to the syntactic investigation of the info code into its segment parts so as to encourage the written work of compilers and mediators. The term may likewise be utilized to depict a part or detachment.

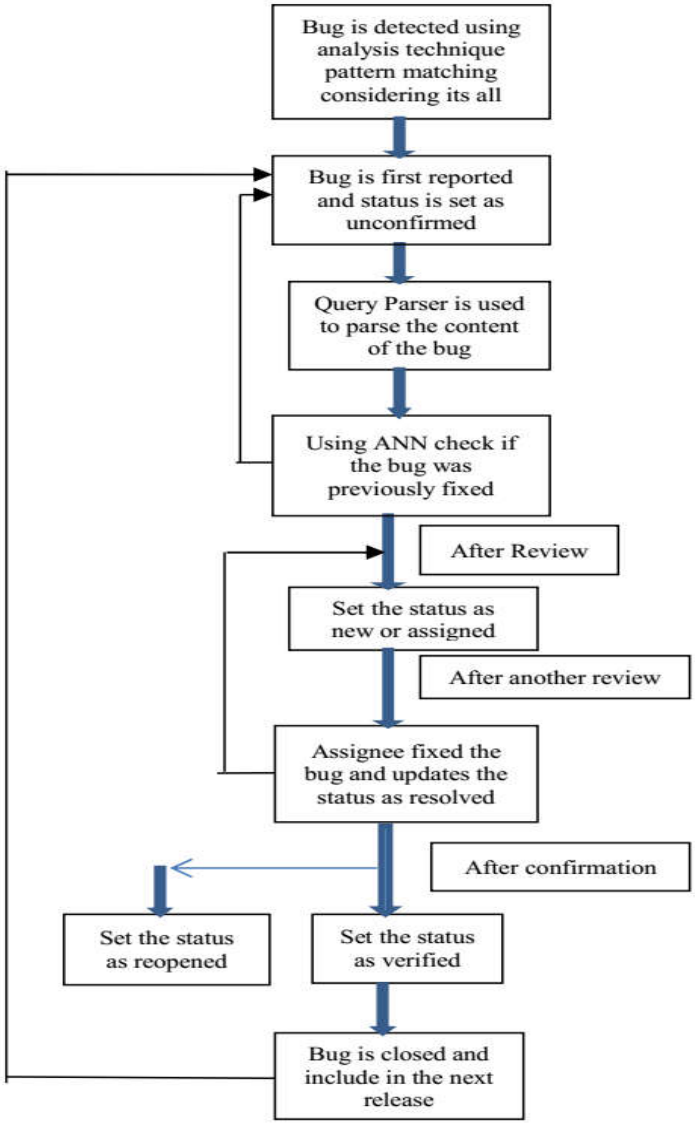

Fig. 1 Architecture of proposed methodology.

Step by step execution of the proposed methodology

Step 1: Bug is detected in the software module using the pattern matching which is an analysis technique,
Step 2: Bug is being reported and status is being assigned as unconfirmed,
Step 3: Query parser is being used to parse the bug into tokens,
Step 4: The tokens and the bugs are then checked using the ANN as they are likely encountered before or not,
Step 5: If yes then solution is being checked and the system goes for another bug, and for no goes for the complete process,
Step 6: Status is assigned as new to the bug and fix the bug and changes the status as verified or reopened,
Step 7: System goes for the next check or next bug.

V. RESULT ANALYSIS

A. *Analysis on the basis of Functionality*

The comparison based on functionality it provide basis shown in a table I. Search and filter, Time Tracking, Usage Statistics, and Automatic Duplicate Bug detection, RSS feed, Localization, Automatic assignment and reassignment of bug to expert.

TABLE 1
ANALYSIS ON THE BASIS OF FUNCTIONALITY

Functionality/ Tool	Search & Filter	Time Tracking	Usage Statistics	RSS Feed	Local-ization	Autom Atic Assig-nment of bug to expert
[4]	Yes	Yes	Yes	Yes	Yes	No
[5]	Yes	Yes	Yes	Yes	Yes	No
[6]	Yes	No	Yes	No	No	No
[7]	Yes	Yes	Yes	Yes	Yes	No
[8]	Yes	No	No	No	Yes	No
[11]	Yes	No	No	No	Yes	No
Proposed Methodo-Logy	Yes	Yes	Yes	Yes	No	Yes

B. Analysis on the basis of Reporting a bug

Reporting a bug in bug tracking system is very important feature of any bug tracking system. A reporting feature should be well designed. We did the analysis of bug tracking tool on the basis of reporting and took the feature online reporting, reporting by e-mail reporting feedback.

TABLE 2
ANALYSIS ON THE BASIS OF REPORTING OF A BUG

Reporting/ Tool	Online	E-mail	Feedback
[4]	Yes	Yes	Yes
[5]	Yes	Yes	Yes
[6]	Yes	No	Yes
[7]	Yes	Yes	Yes
[8]	Yes	Yes	Yes
[11]	Yes	Yes	Yes
Proposed Methodo-logy	Yes	Yes	Yes

VI. CONCLUSION

Bug Tracking System is critical programming normally have tens or hundreds or thousands of imperfections. Bug following framework is use to oversee, settle and organize these deformities. Imperfection following framework is PC database framework that store deformity and help individuals to oversee them. We inferred that present bug following framework have some of restriction. They don't viably gather all the data required by designer, journalist and mysterious client. We have done investigation of bug following framework on the premise of a few criteria. Be that as it may, such paradigm frequently doesn't give wanted outcome. So we propose a new framework of bug tracking system. The propose framework will give an improved level of satisfaction for current bug tracking system. The proposed methodology is better to the existing work on the basis of certain parameters as shown in section result analysis.

REFERENCES

[1] *A. Porter, Jr., Votta, and V. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment", IEEE Transactions on Software Engineering, vol. 21(6), pp. 563- 575, June 1995.*

[2] *H. Guilherme, Travassos, F. S. M. Fredericks Victor, and R. Basili, "Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality", Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Denver, Colorado, 1999.*

[3] *V.B. Singh, and K. K. Chaturvedi, "Bug Tracking and Reliability Assessment System (BTRAS)", International Journal of Software Engineering and Its Applications, vol. 5, No. 4, October, 2011, pp. 17-30.*

[4] *Y. Singh, A. Kaur and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models," Software Quality Journal, vol. 18, no. 3, pp. 3-35, March 2010.*

[5] *F. Khomh, B. Chan, Y. Zou and A. E. Hassan, "An Entropy Evaluation Approach for Triaging Field Crashes: A Case Study of Mozilla Firefox," Reverse Engineering (WCRE), 2011, 18th Working Conference on, Limerick, Ireland, 2011.*

[6] *L. Huang, V. Ng, I. Persing, M. Chen, Z. Li, R. Geng and J. Tian, "AutoODC: Automated generation of orthogonal defect classifications," Automated Software Engineering, vol. 22, no. 1, p. 3–46, March 2015.*

[7] *Y. Tian, D. Lo and C. Sun, "Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction," Reverse Engineering (WCRE), 2012 19th Working Conference on, Kingston, ON, Canada, 2012.*

[8] *H. Hosseini, R. Nguyen and M. W. Godfrey, "A Market-Based Bug Allocation Mechanism Using Predictive Bug Lifetimes," Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on, Szeged, Hungary, 2012.*

[9] *M. Gayathri and A. Sudha, "Software defect prediction system using multilayer perceptron neural network with data mining," International Journal of Recent Technology and Engineering, vol. 3, no. 2, pp. 54-59, May 2014.*

[10] *Y. Tian, D. Lo, X. Xia and C. Sun, "Automated prediction of bug report priority using multi-factor analysis," Empirical Software Engineering, vol. 20, no. 5, pp. 1354-1383, 2014.*

[11] *Lazar, S. Ritchey and B. Sharif, "Improving the Accuracy of Duplicate Bug Report Detection Using Textual Similarity Measures," 11th Working Conference on Mining Software Repositories, Hyderabad, India, 2014.*

[12] *M. Fowller, K. Scott, "UML Distilled: Applying the Standard Object Modeling Language", Addison- Wesley, 1997.*

[13] *P. A. Choudhary, and S. Singh, "Neural Network Based Bug Priority Prediction Model using Text Classification Techniques", International Journal of Advanced Research in Computer Science, vol. 8, No. 5, May-June 2017, pp. 1315-1319.*