ISSN 2348 – 8034
Impact Factor- 3.155

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

A SURVEY ON TECHNIQUES OF DEFECT ANALYSIS FOR SOFTWARE PROJECT MANAGEMENT SYSTEM

Tahira Mahboob*1, Humaraia Abdual Ghafoor*2 Warda Imtiaz3and Mukhtiar Bano4
1,2,3,4 Department of Software Engineering, Fatima Jinnah Women University, Pakistan

ABSTRACT

This paper is a survey on defect Analysis for Software Project Management System and it elaborates that three factors depend on the software project management system these are size, effort and quality.

Good quality results in low defects ratio to be found. In software engineering there is no complete defect prediction for software that finds the defect and quality of the software. Number of defects cannot provide the information of quality. Software projects will make defect removal and prevention much easier in case of earlier defect identification. It is necessary to point out the defects from the given set of projects at the first step, for enhancing the quality of the software process. There are three levels of defect management process which are defect detections, defect analysis and defect prevention. .Defects are identified and fixed at first level. In second level find the root causes of the defect why they are not detected in first level. And the third level is related with the defect prevention from recurring in the future. The main research is that how to produce quality software with minimum number of defects. The main factor of success is the software will be in time and budget.

Keywords: *Defect Prevention, Quality object-oriented, Orthogonal Defect Classification, Defect Management, Defect Analysis, cost constructive model.*

I. INTRODUCTION

Software is developed by human so defect will be remaining in whole life of software. Quality software with the least number of defects to decrease the influence of problems in the organization is basically the main goal of well organized software defect management process. The failure analysis and root-cause analysis in integration are potentially more valuable than subjective assessments, because they quantify defect costs for a specific organization.

Software defect data is most important available management information source, for software process improvement decisions. The researchers were looking for conducts on the other hand; to guide defect correction effort, using information retrieval and natural language processing, to cluster defect reports. Right now, there is no estimation method grew through exact investigation to assess the choice capacity of a task chief towards asset designation for viable deformity administration. The imperfection system is taking into account examining the imperfections that had risen up out of different phases of programming advancement like Requirements, Design, Coding, Testing and Timeline; surrenders because of absence of time amid improvement. The deformity infusion metric quality, once computed, serves as a measuring stick to make an examination in the upgrades made in the product process improvement between comparable arrangements of ventures. Significant disarray wins at the last phases of programming advancement as to which desert found has a place with which period of programming improvement life cycle. Systems like main driver investigation and orthogonal imperfection arrangement are a portion of the normally utilized practices.

II. SOFTWARE PROJECT MANAGEMENT SYSTEM TECHNIQUES

2.*1) On the Value of Static Analysis for Fault Detection in Software*

No single programming shortcoming recognition procedure is equipped for tending to all deficiency identification concerns. Correspondingly to programming surveys and testing, static examination apparatuses or mechanized static investigation can be utilized to uproot imperfections preceding arrival of a product item. To focus to what degree mechanized static examination can help in the monetary generation of a brilliant item, we have investigated static investigation blames and test and client reported disappointments for three huge scale mechanical programming frameworks created at Nortel Networks. The information show that robotized static examination is a moderate method for programming flaw recognition. Utilizing the Orthogonal Defect Classification plan, we found that computerized static investigation is successful at distinguishing assignment and checking flaws, permitting the later programming generation stages to concentrate on more perplexing, utilitarian, and algorithmic shortcomings. A larger part of the imperfections found via robotized static investigation have all the earmarks of being created by a couple of key sorts of developer mistakes and some of these sorts can possibly bring about security vulnerabilities. Factual investigation results demonstrate the quantity of robotized static examination issues can be powerful for recognizing issue modules. Our outcomes demonstrate static examination devices are corresponding to other issue recognition methods for the monetary generation of a superb programming item.

2.2)Research on the Application of Data Mining in Software Testing and Defects Analysis

The high reliability programming is not just one of programming strategy improvement instructing focuses, additionally is the product business advancement vital establishment, this paper condenses the information mining to face the identify of the product validity test, the examination and the specialized angle most current exploration, explained the information mining innovation in the product imperfection test application, incorporating defect test in regularly utilized information mining system, information mining framework and programming testing administration framework. Presented particularly in perspective of the product imperfection's distinctive grouping in view of the association standard's product defect parsing procedure's application, proposed in light of the affiliation principle's product recognize assessment system, the reason for which is to reduction programming imperfections and to accomplish the quick development of programming reliability.

2.3)Orthogonal Defect Classification-A Concept for In-Process Measurements

This paper portrays Orthogonal Defect Classification (ODC), an idea that empowers in-methodology criticism to engineers by separating marks on the advancement process from imperfections. The thoughts are advanced from a before finding that shows the utilization of semantic data from deformities to concentrate reason impact connections in the advancement process. This finding is utilized to add to an orderly structure for building estimation and investigation techniques. This paper characterizes ODC and talks about the fundamental and adequate conditions needed to give criticism to an engineer; shows the utilization of the imperfection sort conveyance to gauge the advancement of an item through a methodology; outlines the utilization of the deformity trigger dispersion to assess the viability and in the end the fulfillment of confirmation methods, for example, investigation or testing; gives test results from pilot undertakings utilizing ODC; opens the ways to a wide mixture of examination strategies for giving viable and quick input in view of the ideas of ODC. Programming is one of the slowest courses of action in empowering new business opportunities, arrangements, or measurements of registering, and is a huge piece of aggregate expense. It has likewise been discovered that the overwhelming reason for framework blackout has moved to programming given that it has not kept pace, in the previous couple of years, with enhancements in equipment or upkeep.

2.4)Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects

To produce high quality object-oriented (OO) applications, a strong emphasis on design aspects, especially during the early phases of software development, is necessary. Design metrics play an important role in helping developers understand design aspects of software and, hence, improve software quality and developer productivity. In this paper, we provide empirical evidence supporting the role of OO design complexity metrics, specifically a subset of the Chidamber and Kemerer suite, in determining software defects. Our results, based on industry data from software developed in two popular programming languages used in OO development, indicate that, even after controlling for the size of the software, these metrics are significantly associated with defects. In addition, we find that the effects of these metrics on defects vary across the samples from two programming languages—C++ and Java. We believe that these results have significant implications for designing high-quality software products using the OO approach.
Index Terms: Object-oriented design, software metrics validation, object-oriented languages, C++, Java.

2.5)A Value-Added Predictive Defect Type Distribution Model based on Project Characteristics

The research identified with programming quality has been centered around displaying leftover deserts in programming so as to gauge programming dependability. Right now, programming building writing still does not have a complete deformity forecast for a product item albeit much work has been performed to foresee programming quality. On the other side, the quantity of deformities alone can't be adequate data to give the premise to arranging quality confirmation exercises and surveying them amid execution. That is, for undertaking administration to be enhanced, we have to anticipate other conceivable data about programming quality, for example, in-methodology abandons, their sorts, etc. In this paper, we propose another methodology for anticipating the circulation of imperfections and their sorts taking into account venture qualities in the early stage. For this approach, the model for forecast was made utilizing the bend fitting technique and relapse investigation. The greatest probability estimation (MLE) was utilized as a part of fitting the Weibull likelihood thickness capacity to the real abandon information, and relapse examination was utilized as a part of recognizing the relationship between the task qualities and the Weibull parameters. The exploration model was approved by cross-approval.

2.6)Classification and evaluation of defects in a project retrospective

One of the advantages of defect prevention is the beneficial effect it may have on interval: at the start of process, good quality results in low defects ratio to be found and corrected in the later phases of the process, thus causing an indirect reduction of intervals. As there are three inter-dependent factors that controls project's development processes; interval, quality and cost. The market pressures continue to demand agility in new features; the challenge to project management is to meet those demands while not sacrificing the quality. The root-cause analysis is the core of this threefold study. This paper discuss the novel approach that have been used for defect and root cause classification, in addition to that the mechanisms used for randomly selecting the MRs; to analyze analyses via a web interface. The paper presents the results of analyses of the MRs and explains the defects and root causes that were found. Then the researcher concludes the report on the root-cause analysis. The project context of this study is a network element: a flexibly configurable transmission system in an optical network. The limitations of this paper are: the root-cause study focuses on defect analysis and on determining the underlying root causes of the defects.

2.7) Defect Handling In Software Metrics

In various software projects defect handling is one of the dominant and important activities involved. Software projects will make defect removal and prevention much easier in case of earlier defect identification. It is necessary to point out the defects from the given set of projects at the first step, for enhancing the quality of the software process; then it involves classification and analysis of the pattern and further it involves deduction for defects prevention. The experiences from the past are usually taken by software engineers to prevent the defects from re-occurring. The undesirable aspects of the software quality are generally represented as a defect. In this research paper the study about the various types of defect prevention techniques is analyzed, and then researchers have undergone through the survey of a two-step software defect prediction model for improving the quality of software: COQUALMO cost constructive model. Better software results are produced when early identification of the defect is introduced.

2.8)Software Failure Analysis for High-Return Process Improvement Decisions

To determine the weaknesses in development processes and to decide what changes are needed; the valuable tools in enabling organizations have become: the software failure analysis and root-cause analysis. Transfer process learning from individuals to organizations is a useful way to evaluate software defect. It includes analyzing software defects, also brainstorming the root causes of those defects and incorporating what is learned during training and process changes; so that the defects never occur again. The failure analysis and root-cause analysis in integration are potentially more valuable than subjective assessments, because they quantify defect costs for a specific organization. Software defect data is most important available management information source, for software process improvement decisions. This paper elaborates the aspect that how software defects data is influential management in sequence source. Using it effectively will help attain an optimal balance between respond to defect information, and proactively taking decisions toward thwarting future defects.

2.9) Automatic Defect Classification: An Experience Applying Natural Language Processing

This paper shows early results of a research effort that merges two approaches on Quality Management; on one hand, working with process development initiatives involving defect underlying analysis, and the problem of measuring inter-rater trustworthiness for defect classification taxonomy has occurred in a software development organization.

The researchers were looking for conducts on the other hand; to guide defect correction effort, using information retrieval and natural language processing, to cluster defect reports. The value of the fact that a software development organization can improve from their defects nevertheless, the analysis of each individual defect can be tiresome, error prone, and time taking, has already been established by previous research. As a proxy to estimate remaining defect density in delivered software a practical application can also include the use of defect clustering combined with contextual information from the organization. As mentioned earlier, clustering can also be used to prioritize which defect to cater first.

2.10)Modelling and Analysing of Software Defect Prevention Using ODC

As the time passes the product intricacy is expanding and because of this product unwavering quality and quality will be influenced. Furthermore, for measuring the product dependability and quality different imperfection estimation and deformity following instrument will be utilized .Software imperfection anticipation work normally concentrates on individual examination and testing system. ODC is a component by which we abuse programming desert that happen amid the product improvement life cycle. Orthogonal deformity arrangement is an idea which empowers engineers, quality directors and task chiefs to assess the viability and rightness of the product. Programming imperfection avoidance is a vital piece of the product advancement. The quality; dependability and the expense of the product item intensely rely on upon the product imperfection location and counteractive action process. In the advancement of programming item 40% or a greater amount of the task time is spent on deformity discovery exercises. Programming deformity anticipation exploration has proposed new investigation and testing strategies and has concentrated on and analyzed diverse assessment and testing systems. In the paper the fundamental thought is to give execution of ODC in certifiable application. It starts with and review of different deformity arrangement plans took after by ODC ideas. The last part will depict how we will embrace ODC in programming improvement. The end of this paper portrays the Improvement in programming venture in the wake of executing ODC. Page Layout a simple approach to agree to the meeting paper designing necessities is to utilize this report as a format and basically sort your content into it.

2.11) Defect prevention based on 5 dimensions of defect origin

The expanding reliance of society on programming and the critical results that a disappointment can result in obliges the need to discover the deformities at the beginning itself. In view of the lessons learnt from the prior arrangement of activities, a deformity structure highlighting the 5 Dimensions (Ds) of imperfection starting point is proposed in this work. The imperfection system is taking into account examining the imperfections that had risen up out of different phases of programming advancement like Requirements, Design, Coding, Testing and Timeline; surrenders because of absence of time amid improvement. This study is not constrained to simply recognizing the inception of imperfections at different periods of programming advancement additionally figures out the explanations behind such surrenders, and deformity preventive (DP) measures are proposed for every sort of imperfection. This work can help specialists pick compelling deformity shirking measures. Notwithstanding touching base at imperfection structure, this work additionally proposes a deformity infusion metric taking into account seriousness of the deformity instead of simply surrender check, which gives the quantity of balanced deformities created by an undertaking at different stages. The deformity infusion metric quality, once computed, serves as a measuring stick to make an examination in the upgrades made in the product process improvement between comparable arrangements of ventures. Significant disarray wins at the last phases of programming advancement as to which desert found has a place with which period of programming improvement life cycle. Systems like main driver investigation and orthogonal imperfection arrangement are a portion of the normally utilized practices.

2.12) Establishing a Defect Management Process Model for Software Quality Improvement

Software is developed by human so defect will be remaining in whole life of software. Quality software with the least number of defects to decrease the influence of problems in the organization is basically the main goal of well organized software defect management process. There are three levels of defect management process which are defect detections, defect analysis and defect prevention .Defects are identified and fixed at first level. In second level find the root causes of the defect why they are not detected in first level. And the third level is related with the defect prevention from recurring in the future. The main research is that how to produce quality software with minimum number of defects. The main factor of success is the software will be in time and budget. In this research paper, establish a defect management process model that reduce number of defects and provide a quality software product. Mostly Organizations are using ITIL defect management process model. The major challenges in the ITIL defect management process model are lack of performance and less participation of customer in process.

2.13) Better Management of Defects for Improving Software Processes

Testing is very important after developed software because no software can build defect free software. Defect Tracking System is the tool that used for reporting defects after testing software. Software quality enhanced by defects reported. In this research paper research is on how defects are managed and approaches that are used for managing defects and for improvement that will prevent the defects in future. Defect is basically erroneous information in software. This erroneous information is due to an error in design, specification and mostly in requirement. These defects and errors are finding in testing and identified during reviews because these defects creates complexity in software. Defect defined as it is unplanned event that occurs during reviews and in testing. These defects require investigation and correction and these are created when results are different from expected results. When defect is indentified then its related information is recorded and this information is called Defect report .Developer review the defect report and try to resolve.

2.14) Myths and Strategies of Defect Causal Analysis

Causal analysis activities are Sigma, CMMI, and Lean and these are popular process improvement approaches. The concept of causality technique is used in causal analysis which is misunderstood and misapplied. For defect causal analysis many different processes, tools, and techniques are developed now-a- days. These different processes, tools, and techniques are successful in many situations. Many Organizations invest large amounts in software and training for deploy them. The application of these techniques provides sources of problems. For correct identification and effective decision of "deep" problems, a checklist implementation of the techniques is not enough. In the software industry it is becoming more common and focuses on increasing quality improvement. It is important training attentive planning is used to ensure for successful implementation. This research paper finds the major problems and suggests the solution of these problems.

2.15) A Case Study in Root Cause Defect Analysis

Interval, quality and cost are three interdependent factors that make our software development processes. New features rapidly demand in market without any defect and with high quality. An upstream quality improvement Practice is one advantage of defect prevention. Indirect interval reduction produced by higher quality early in the process results in fewer defects to be found .the defect Modification Requests (MRs) revealed by root cause defect analysis study. In this research paper defect and root cause classification and the mechanisms and used for arbitrarily selecting the MRs to examine and collecting the analyses via a web interface. 5-10 people teams in the NE software are developed.NE configuration consists of different hardware board types and equal to 150 different software components. Subsystem is the architectural unit that a software team is responsible for a collection of functionally related components, which altogether form an architectural unit. The mission of the RCA project was find systematic root causes of defects, during the maintenance release analyze major customer reported MRs, reduce number of critical defects in project.

2.16) Defect Analysis and Prevention for Software Process Quality Improvement

"An ounce of aversion is justified regardless of a pound of cure." In programming, these interpretations interpret into the normal perception that the longer a deformity stays in process, the more lavish it is to alter besides programming deformities are extravagant and time expending. The expense of discovering and adjusting imperfections speaks to a standout amongst the most lavish programming improvement exercises. What's more, that as well, if the slips escape till the last acknowledgement testing phase of the venture life cycle, then the undertaking is at a more noteworthy hazard regarding now is the ideal time and Cost variables. A little measure of exertion spent on quality affirmation will see great measure of expense reserve funds in terms of distinguishing and dispensing with the imperfections. To pick up a more profound comprehension of the adequacy of the product process, it is fundamental to look at the points of interest of imperfections distinguished in the past undertakings and to study how the same can be dispensed with due to process enhancements and more up to date approaches. This paper will concentrate on discovering the aggregate number of imperfections that has happened in the product improvement process for five comparable ventures and goes for arranging different deformities utilizing first level of Orthogonal Deformity Classification (ODC), discovering underlying drivers of the deformities and utilize the learning of the tasks as preventive thoughts. The paper too showcases on how the preventive thoughts are executed in another set of ventures bringing about the decrease of the quantity of comparable deformities.

2.17) Handling of Software Quality Defects in Agile Software Development

Programming quality affirmation is concerned with the effective and powerful advancement of vast, dependable, also, top notch programming frameworks. In coordinated programming advancement and support, refactoring is a vital stage for the constant change of a product framework by uprooting quality deformities like code scents. As time is critical figure coordinated advancement, not all quality deformities can be evacuated in one refactoring stage (esp. in one emphasis). Documentation of value surrenders that are found amid robotized or manual revelation exercises (e.g., pair writing computer programs) is important to evade exercise in futility by rediscovering them in later stages. Lamentably, the documentation and treatment of existing quality deserts and refactoring exercises is a typical issue in programming support. To review the reasons why changes were done, data must be separated from either exclusive documentations or programming forming frameworks. In this part, we depict a procedure for the repeating what's more, practical disclosure, taking care of, and treatment of value deserts in programming frameworks. An annotation dialect is exhibited that is utilized to store data about quality deformities found in source code and that speaks to the deformity furthermore, treatment history of a piece of a product framework.

III. ANALYSIS DESCRIPTION

Results of analysis of evaluation techniques are shown in the table 1.

Many techniques are discussed against defect analysis. Important techniques are discussed in the first half and rest of them is discussed in the later. In Table 1 all the techniques along with their Authors and research paper while table 2 and table 3 show the definition of the defect types and defect type of the software . There are many techniques discuss in research papers like Orthogonal Defect Classification plan, information mining framework and programming testing administration framework, Orthogonal Defect Classification, Design metrics, bend fitting technique and relapse investigation, The root-cause analysis,COQUALMO cost constructive model, defect clustering combined with contextual information, Orthogonal deformity arrangement, deformity preventive (DP) measures, ITIL defect management process model, Defect Tracking System, Causal analysis activities are Sigma, CMMI, and Lean, defect Modification Requests (MRs),Refactoring in table 1. In table 2 there are some defect type are summarized on the basis of research and discuss in research papers techniques. In table 3 defect types are discuss which are analyze in software products these are Algorithm,Documentation,Checking,Assignment,FunctionInterface,Requirements,Timing ,Logical Error, Graphical error, Design error. Defect types are in table 3 which defined quality of the software and defect produced in which phase,

IV. CONCLUSION

This research paper includes different defect prevention techniques and analyses these techniques critically. Defect defined as it is unplanned event that occurs during reviews and in testing. These defects require investigation and correction and these are ,created when results are different from expected results. The project context of this study is a network element: a flexibly configurable transmission system in an optical network. Mostly Organizations are using ITIL defect management process model. The major challenges in the ITIL defect management process model are lack of performance and less participation of customer in process.The limitations of this paper are: the root-cause study focuses on defect analysis and on determining the underlying root causes of the defects. In the software industry it is becoming more common and focuses on increasing quality improvement. It is important training attentive planning is used to ensure for successful implementation. This research paper finds the major problems and suggests the solution of these problems.

REFERENCES

1. *J. Zheng, L. Williams,N. Nagappan, W. Snipes, J. P. Hudepohl, and M. A. Vouk. "On the Value of Static Analysis for Fault Detection in Software" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 32, NO. 4, APRIL 2006. [Available] http://collaboration.csc.ncsu.edu/laurie/Papers/TSE-0197-0705-2.pdf*
2. *Y. Shen, J. Liu. "Research on the Application of Data Mining in Software Testing and Defects Analysis", 2009 Second International Conference on Intelligent Computation Technology and Automation. [Available] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5287758*
3. *R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M-Y.Wong. "Orthogonal Defect Classification-A Concept for In-Process Measurements" IEEE TRANSACTTONS ON SOFTWARE ENGINEERING, VOL 18, NO. 11, NOVEMBER 1992 [Available] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=177364*
4. *R. Subramanyam, M.S. Krishnan. "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29, NO. 4, APRIL 2003 [Available] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1191795*
5. *Y. Hong, J. Baik, I-Y. Ko, H-J. Choi. "A Value-Added Predictive Defect Type Distribution Model based on Project Characteristics" Choi Seventh IEEE/ACIS International Conference on Computer and Information Science [Available]*
6. *M. Leszaka, D E. Perryb, D. Stolla. "Classification and evaluation of defects in a project retrospective." The Journal of Systems and Software Volume 61, (2014). http://users.ece.utexas.edu/~perry/work/papers/DP-02-jss.pdf*
7. *Mittal1, S K. Dubey. "Defect Handling In Software Metrics." ISSN : 2278 – 1021 International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, (May 2012). http://www.ijarcce.com/upload/may/Defect%20Handling%20Using%20COQUALMO%20Defect%20Prediction%20Model.pdf*
8. *R B. Grady. "Software Failure Analysis for High-Return Process Improvement Decisions." Hewlett-Packard Journal August 1996. Article 2 http://www.hpl.hp.com/hpjournal/96aug/aug96a2.pdf*
9. *S. Matalonga, T S. Feliu, V. Rus. "Automatic Defect Classification: An Experience Applying Natural Language Processing." Dunn Hall, Memphis TN. Department of Computer Science, University of Memphi.(2011) http://fi.ort.edu.uy/innovaportal/file/2038/1/automaticdefectclasification.pdf*
10. *P. Trivedi, S. Pachori. "Modelling and Analysing of Software Defect Prevention Using ODC." International Journal of Software Engineering & Applications(IJSEA), Vol.3, No.4, (July 2012) http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.205.9265*
11. *S. Kumaresh, B. Ramachandran. "Defect Prevention Based On 5 Dimensions Of Defect Origin" (2013) http://airccse.org/journal/ijsea/papers/3412ijsea07.pdf*
12. *H. A. Khan. "Establishing a Defect Management Process Model for Software Quality Improvement." Internet: www.ijfcc.org/papers/232-B10100.pdf, 6, December 2013.*
13. *S.Mittal1 , K.Solanki2, A. Saroha3. "Better Management of Defects for Improving Software Processes." Internet: journaldatabase.info/download/pdf/better_management_defects_for Improving Software Processes.pdf, 02, Aug 2011.*
14. *D.N.Card. "Myths and Strategies of Defect Causal Analysis." Internet: www.researchgate.net/...Myths_and_strategies_of_defect_causal_analysis.pdf, 22 October 2006.*
15. *M.Leszak, D.E.Perry & D.Stoll. "A Case Study in Root Cause Defect Analysis." Internet: users.ece.utexas.edu/~perry/work/papers/DP-00-icse-rca.pdf, 7, Nov 2011.*
16. *S. Kumaresh & R. Baskaran. "Defect Analysis and Prevention for Software Process Quality Improvement." Internet: ijcaonline.org/volume8/number7/pxc3871759.pdf, .7, October 2010.*
17. *J.RECH. "Handling of Software Quality Defects in Agile Software Development." Internet: joerg rech.com/Paper/BC_Rech_AgileQDH_final.pdf , Oct. 25, 2008*

TABLE 1 DEFECT TECHNIQUES IN RESEARCH PAPER

Serial#	*Author*	*Research paper*	*Techniques*
1	***J. Zheng, L. Williams,N. Nagappan, W. Snipes, J. P. Hudepohl, and M. A. Vouk***	*On the Value of Static Analysis for Fault Detection in Software*	*Orthogonal Defect Classification plan*
2	***Y. Shen, J. Liu***	*Research on the Application of Data Mining in Software Testing and Defects Analysis*	*information mining framework and programming testing administration framework*
3	***R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M-Y.Wong***	*Orthogonal Defect Classification-A Concept for In-Process Measurements*	*Orthogonal Defect Classification*
4	***R. Subramanyam, M.S. Krishnan***	*Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects*	*Design metrics*
5	***Y. Hong, J. Baik, I-Y. Ko, H-J. Choi***	*A Value-Added Predictive Defect Type Distribution Model based on Project Characteristics*	*bend fitting technique and relapse investigation*
6	***M. Leszaka, D E. Perryb, D. Stolla***	*Classification and evaluation of defects in a project retrospective*	*The root-cause analysis*
7	***Mittal1, S K. Dubey***	*Defect Handling In Software Metrics*	*COQUALMO cost constructive model*
8	***R B. Grady***	*Software Failure Analysis for High-Return Process Improvement Decisions*	*The root-cause analysis*
9	***S. Matalonga, T S. Feliu, V. Rus***	*Automatic Defect Classification: An Experience Applying Natural Language Processing*	*defect clustering combined with contextual information*
10	***P. Trivedi, S. Pachori***	*Modelling and Analysing of Software Defect Prevention Using ODC*	*Orthogonal deformity arrangement*
11	***S. Kumaresh, B. Ramachandran***	*Defect Prevention Based On 5 Dimensions Of Defect Origin*	*deformity preventive (DP) measures.*
12	***H. A. Khan***	*Establishing a Defect Management Process Model for Software Quality Improvement*	*ITIL defect management process model*
13	***S.Mittal[1] , K.Solanki[2] , A. Saroha[3]***	*Better Management of Defects for Improving Software Processes*	*Defect Tracking System*
14	***D.N.Card***	*Myths and Strategies of Defect Causal Analysis*	*Causal analysis activities are Sigma, CMMI, and Lean*
15	***M.Leszak, D.E.Perry & D.Stoll***	*A Case Study in Root Cause Defect Analysis*	*defect Modification Requests (MRs)*
16	***S. Kumaresh & R. Baskaran***	*Defect Analysis and Prevention for Software Process Quality Improvement*	*Orthogonal Deformity Classification (ODC)*
17	***J.RECH***	*Handling of Software Quality Defects in Agile Software Development*	*Refactoring*

TABLE 2 DEFINATION OF DEFECT TYPE ANALYSIS IN RESEARCH PAPER

Serial#	*Defect Type*	*Meanings*	*Possible Values*
1.	**Algorithm**	A programming algorithm is a set of instructions often created as functions, designed to perform a specific task.	Yes, No
2.	**Documentation**	The product's technical manuals and online information that provide the information that describes the product to its users is documentation.	Yes, No, Not defined
3.	**Checking**	On occurrence of some hardware or software problem, a problem report is created so one can check for a solution.	Yes, No, Not defined
4.	**Assignment**	Assigning of tasks of a project to eligible team members.	Yes, No
5.	**Function**	A group of statements in a software program that combine to perform a task.	Yes, No
6.	**Requirements**	The software requirements are comprehensive description of what the software will do and how it will perform.	Yes, No
7.	**Timing**	The response time of a software product, how efficient it is.	Yes, No
8.	**Interface**	It enables different programs to communicate with each other and the operating system; GUI enables a user and a computer to communicate with each other.	Yes, No
9.	**Logical Error**	It is a bug in a program that produces undesired output or causes it to operate incorrectly.	Yes, No
10.	**Graphical error**	Error in user interface (GUI) of a software product.	Yes, No
11.	**Design error**	Error in sketch, architecture or in plan of a software product.	Yes, No, Not defined

NOTE: Y=YES, ND=NOT DEFINED, N=NO

ISSN 2348 – 8034
Impact Factor- 3.155

TABLE 3 ANALYSIS OF DEFECT TYPE AGAINST AUTHORS

Serial #	***Author***	***Algorithm***	***Documentation***	***Checking***	***Assignment***	***Function***	***Interface***	***Requirements***	***Timing***	***Logical Error***	***Graphical error***	***Design error***
1	***J. Zheng, L. Williams,N. Nagappan, W. Snipes, J. P. Hudepohl, and M. A. Vouk***	Y	Y	Y	Y	N	Y	Y	N	Y	N	N
2	***Y. Shen, J. Liu***	Y	Y	N	Y	Y	Y	N	Y	Y	N	Y
3	***R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M-Y.Wong***	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y
4	***R. Subramanyam, M.S. Krishnan***	N	ND	Y	N	Y	Y	N	Y	Y	N	Y
5	***M. Leszaka, D E. Perryb, D. Stolla***	Y	Y	N	Y	Y	Y	N	Y	Y	N	Y
6	***Y. Hong, J. Baik, I-Y. Ko, H-J. Choi***	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Y
7	***Mittal1, S K. Dubey***	N	Y	N	Y	Y	N	Y	N	Y	Y	Y
8	***R B. Grady***	N	Y	N	Y	N	Y	N	Y	Y	N	Y
9	***S. Matalonga, T S. Feliu, V. Rus***	N	Y	N	Y	Y	Y	Y	Y	Y	Y	ND
10	***P. Trivedi, S. Pachori.***	Y	Y	N	Y	N	Y	Y	Y	N	N	Y
11	***S.Kumaresh,B.Ramachandran***	Y	Y	Y	Y	Y	N	Y	N	N	Y	Y
12	***H. A. Khan***	Y	Y	Y	Y	N	Y	Y	Y	Y	N	N
13	***S.Mittal , K.Solanki,A.Saroha***	Y	Y	N	N	Y	Y	ND	Y	Y	N	Y
14	***D.N.Card***	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
15	***M.Leszak, D.E.Perry & D.Stoll***	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Y

ISSN 2348 – 8034
Impact Factor- 3.155

16	***S.Kumaresh &R.Baskaran***	Y	Y	N	N	Y	N	N	Y	N D	Y	Y
17	***J.RECH***	Y	Y	Y	Y	Y	N	Y	N	N	Y	Y