International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-7 Issue-4, September 2018

Analysis of "Defect Root Cause and Corrective Actions" and its impact on "Software System Quality"

Abhishek Anurag, R. Kamatchi

Abstract: ***Nature of software systems and its usage have changed significantly, so its complexity has grown exponentially. It is very difficult and challenging to maintain quality of software systems as user requirements and environment in which system is being used are constantly changing. If user requirements are not met, it is defined as defect. Defect must be analyzed, root caused and fixed to help improve software system quality. In this research study defects are analyzed based on root cause analysis, and different root cause categories and corresponding corrective actions are identified. This study analyses different attributes of defects which are interdependent with root cause & corrective actions and how they impact software system quality.***

Keywords: Software Quality; Defect Attributes; Root Cause Analysis; Corrective Actions

I. INTRODUCTION

A 'Software System' consists of multiple different programs, configuration files explaining how to configure these programs, system documentations giving information of the structure of the system, user documentations on how customers will use the system and web sites for downloading the software and documentations [1]. Software systems have become very complex due to nature and usage of these systems. So, maintaining quality of such complex and ever-changing software systems are challenging, complex and costly. Quality is having numerous definitions and few among them are [2]:

1. Quality consists of those software system features which meets user's requirements
2. Quality consists of freedom from deficiencies or called as "fitness to use".

If software system is not meeting user's need or it is not fit to use or having deficiencies, then quality is considered as poor. Even as per ISO 9000, quality is the degree to which a commodity meets the requirements of the customer at the start of its life. These deficiencies or issues are called defect in Software Engineering.

Revised Manuscript Received on 30 September 2018.
***** Correspondence Author
Abhishek Anurag*, Research Scholar, Department of Computer Science and Engineering, Amity School of Engineering & Technology, Amity University Mumbai (Maharashtra), India.
R. Kamatchi, Head, Department of Computer Science and Engineering, Amity School of Engineering & Technology, Amity University, Mumbai (Maharashtra), India.

Defects play crucial role in help improving customer experiences. Lesser the defects, better the experiences. So, defect detection and fixing them is very critical. To avoid its recurrence defects are analyzed, root caused, and corrective actions are taken to improve the quality. One of most important defect analysis technique is RCA – root cause analysis of defects [3].

A case study is done on defects of Company X, Product X & Release X to understand root cause of defects, corrective actions of defects and quality of software system and analyze the relation between them.

1.1. Software System Overview

Analysis of software quality is based on "Product X" of "Company X", which is one of the world's most advanced streaming device. This is a very complex Android based embedded software system which is under analysis. The software build released to customer over the air (OTA) here is called "Release X". It took almost two years for this software system to launch into the market. 761 engineers were assigned on this product and out of these 70 were from test engineering team. This software system was having 926 software components, 588K numbers of files, 241 million lines of code (LOC) and 38 million lines of code (LOC) as statements. Overall this product had 6 major and 10 minor public releases to customers which was over the air (OTA). For the span of four years for Product X almost 21K defects were logged into defect database management and were analyzed. Also, almost 23K tests available in test database management for Product X were analyzed [4, 5].

1.2. Objective of this Analysis

As part of this study - defects, root cause and corrective actions of defects are analyzed to understand its impact on quality of software system. For analysis of defects one of root cause analysis techniques called RCA [6] was followed. Goal of this analysis are:

1. Based on RCA techniques analyze 21K defects of Product X.
2. Define different categories of defect's root cause and associated corrective actions.
3. Define all attributes of defects which are associated with root cause and corrective actions.
4. Analyze all tests of Product X and define different test attributes.
5. Find interdependencies of defect and test attributes.
6. Define how software quality is dependent on all identified attributes of defects and tests.

1.3. Methodologies

This study is based on experimental research or can be said is causal research. The dependent variable in this study is "Software Quality" and independent variables are based on defects and tests attributes. These attributes are based on analysis based on defect's root cause and corrective actions. For data collection "purposive sampling" technique is used. Also, this study is based on methods of "convenience sampling" techniques. Data is collected from Company X and Product X. Data is collected from "defect database management" and "test database management" systems. These data are exported into Microsoft Excel 2016 and based on Excel macros, mathematical functions and formulas, all data processing, calculations and analysis of data has been done and conclusions are derived.

1.4. Related Work

This analysis is extension of below two studies:

1. Study on Software Quality improvement based on Root Cause & Corrective Actions [5].
2. A Case Study of existing Quality Model based on Defects & Tests Management of Embedded Software System [4].

First study [5] is mostly a kind of literature review studies. Customer defects are something which impacts software quality and reduces the customer experiences. If customer defects are fixed, quality of the software product improves. In this study it is focused on defects root cause and their corrective actions. Also, how different root causes and corrective actions if taken systematically will help improve software quality.

Second study [4] is mainly about analysis of all customer defects and tests of an Android based embedded software product from a company. Here every attribute of defects are tests are analyzed. Based of root cause and corrective actions, a co-relation and interdependencies of defects and tests are defined in this study. Also, what is existing quality model and on what parameters is depends, it is analyzed in this study.

Part of both studies major focus is on: defects, defect attributes, defect's root cause, different corrective actions, test and test attributes and co-relation between them. These were independent variables and dependent variable they are impacting which is under study is "Software Quality".

II. DEFECT DATA ANALYSIS

Defect data for Product X and Release X were pulled and analyzed for different origins. It was observed that majorly 17 different teams were there who contributed towards defects. Different teams and number of defects they detected can be seen in [Figure 1]:

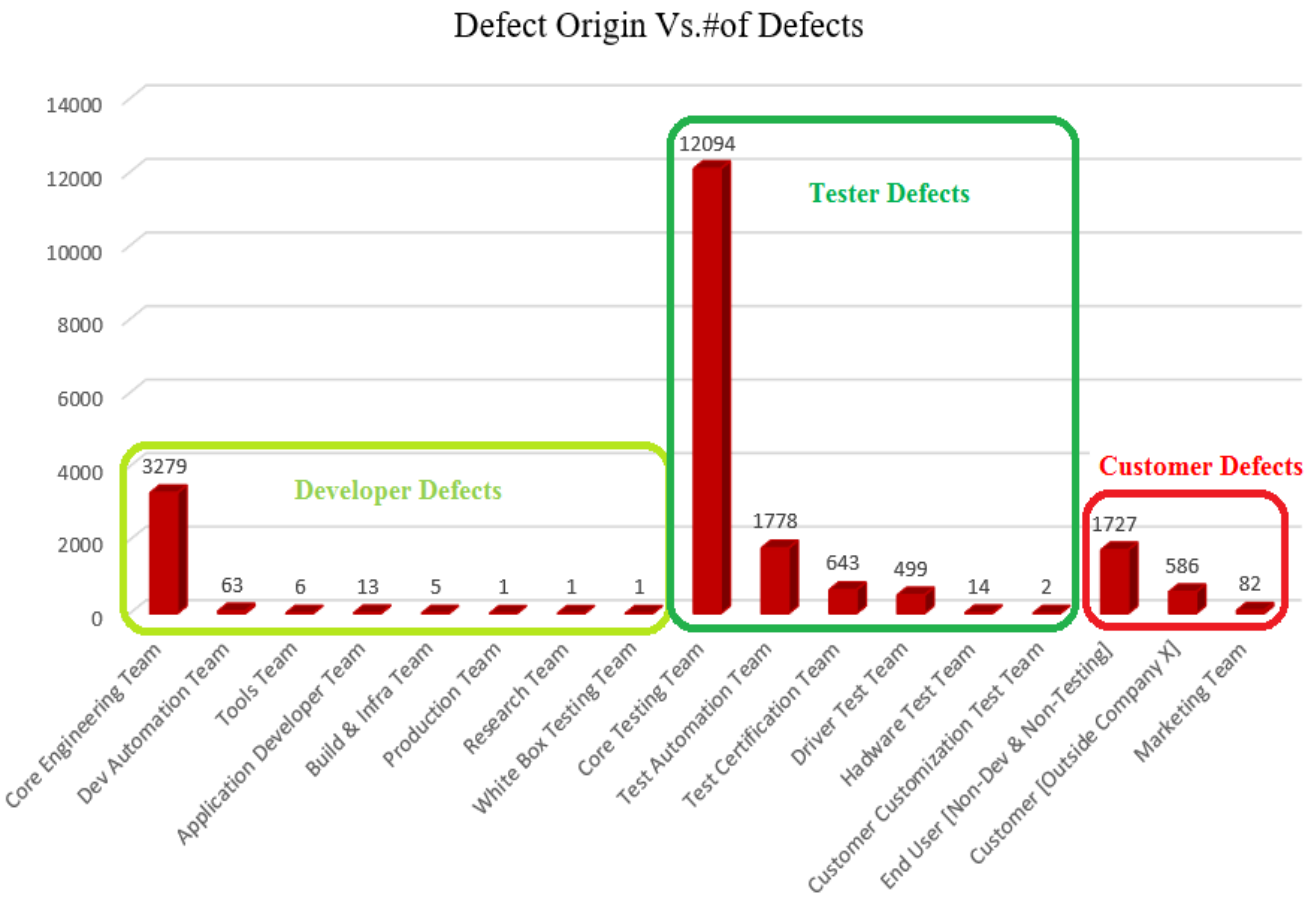

Fig. 1: Defect Origin - Different Teams

After analysis it is observed that majority of defects are caught during validation by testers or developers. 72.28% defects were caught by testers while doing validation at different stage of software system. 16.35% defects were caught by developers while design, development and their validation of the software system. Only 11.52% defects were not caught by either tester or developer [Figure 2].

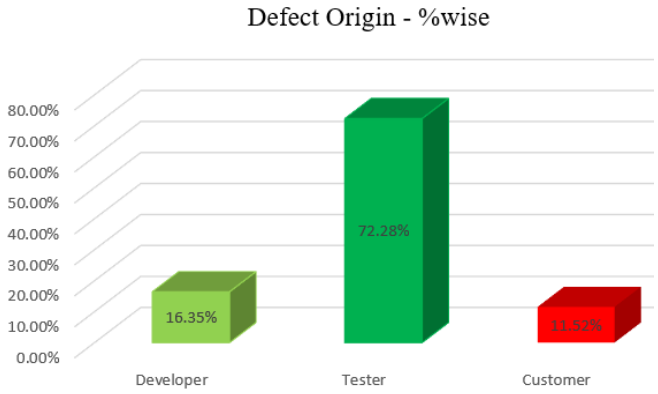

Fig. 2: Defect Origin Distribution

Since customer defects were caught by users, it caused poor user experiences and hence quality of the product is considered poor. To help improve the quality customer defects are needed to be analyzed, root cause and fixed. Once defect is logged the first step is to understand the impact of the defects. This is defined by defect severity. Once defect is originated, then severity or its impact need to determine based on customer experiences. Before categorizing all defects into different severity bucket, different level of severity is identified and defined. There were different seven level of severity was defined based on software system under study. These were: 1. Functionality – delivered features are not working. 2. System Crash – device is not responsive. 3. Application Crash – applications are not responsive. 4. Corruption – audio video quality is poor. 5. Performance – device or applications are very slow. 6. Enhancement – new wishlist feature request. 7. Task Tracking – tracking actions related to defects. After these severity categories was identified and defined, all defects were analyzed and categorized based on this [Figure 3].

Fig. 3: Defect Severity Distribution

After analysis it was observed that 6.84% of times functionality was broken, 0.63% of times applications were crashing, 0.48% of times system was crashing and 0.41% of times corruptions were seen by customers while using the software system. Since defect origin and severity is defined, it was further analyzed whether defect is a regression. A defect is a regression if it is logged because any existing feature or functionality which was working earlier is now not working on latest software build. If a defect is logged for first time this is called as "Not a Regression (NAR)". If a defect is not determined as regression and NAR, they fall into categories undermined. All defects in this case study were analyzed for regression as this help defines what is priority of the defects to be fixed [Figure 4].

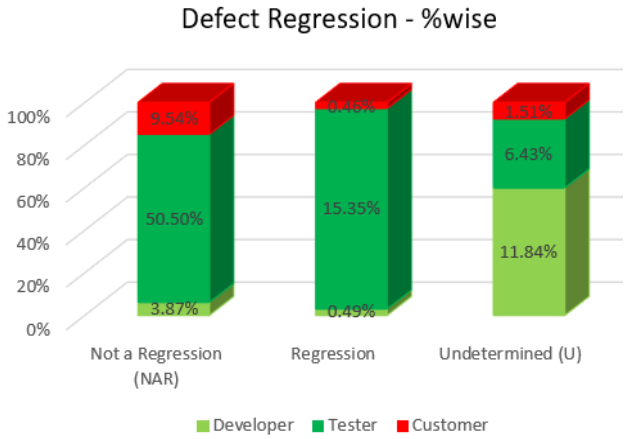

Fig. 4: Defect Regression Distribution

After categorization and analysis, it is observed that 0.46% customer defects were logged as working functionality or features were broken. 9.54% defects were seen for the first time by customers. 1.51% defects were not certain that customer is seeing for the first time or not. Based on origin, severity and regression of defects, its priority is defined. Priority is something which tells in which order these defects must be fixed. Defects whose impact are very high on quality are also prioritized as fixing lower quality bugs won't help improve user experiences significantly. Based on this case study first different categories of priority were defined. It is as: 1. P1 – called showstopper. These defects can block the release made to customers. 2. P2 – called as 'fix before next build'. These defects annoy all users and they follow on this very frequently. 3. P3 – called as 'fix before next milestone'. These defects are less annoying and not followed frequently. 4. P4 – called 'nice to have'. These are based on new feature request from customers. 5. U – called unprioritized, which are not that important defects, and no one follows on these. Based on these categories all defects were analyzed and mapped with priorities defined in [Figure 5].

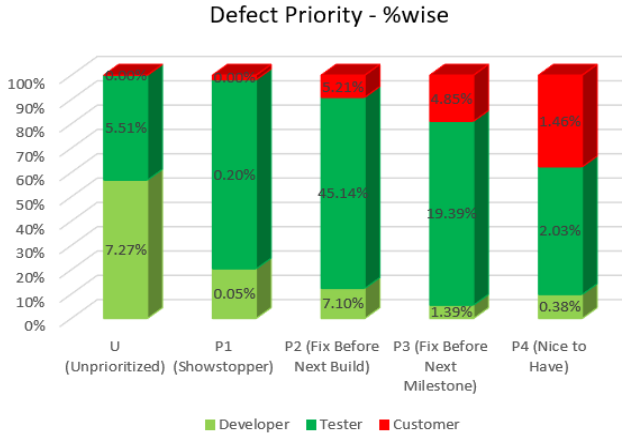

Fig. 5: Defect Priority Distribution

From customer point of all defects were prioritized and observed that none of defects reported were showstopper in nature. 5.21% defects were logged by customer which needs to be fixed in next software
build getting released as extensively followed by customers. 4.85% defects were of those categories which were prioritized for next major milestone as no customer was escalating issues on these. 1.46% customer defects were prioritized as wish list for new feature request. Once all defects were defined and categorized with severity, regression and priority, it was further analyzed for disposition. Defects are analyzed in this case study to know what is its final state. The defect dispositions defined are: 1. Defect fixed – here root cause of defect was found, fixed and deployed in software system. 2. Not a Defect – these are defects as per design and customers are not aware of these functionality and features. 3. Defect will not Fix – due to technical and business reasons these defects are decided not to be fixed. 4. Defect still not Fixed – root cause of defect is still not known, so no fix provided yet. 5. Defect existing (Duplicate) – root cause of defect is same as another defect already logged. 6. Defect third party – other organization needs to root cause and fix the defect as it is part of their applications and systems. 7. Defect fix not known – Exact root cause is not known. Fix provided in another defect caused this defect to be fixed. Based on these definitions all defects were analyzed and categorized [Figure 6].

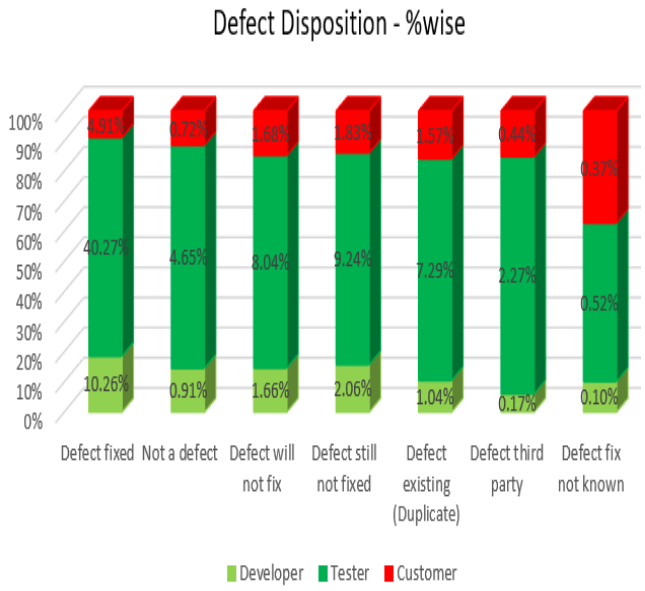

Fig. 6: Defect Disposition Distribution

Once defect disposition was categorized, defect was further analyzed for its root cause based on fixes provided in the defect itself. For RCA analysis team of subject matter experts were formed. Members of these teams were from different defect teams and software system teams, viz. defect origin teams, subject matter experts from software module, technical product managers, program managers, architects, sales and marketing team members, hardware teams, etc. For every defect a group of experts from each team was picked based on not only defect area but also based on nature of the problem and the fixes provided. A root cause and corrective action template was designed to ensure the root cause process is followed to define different root cause and corrective actions based on software system under study. Summary of the RCCA template can be seen into [Figure 7]:

Root Cause & Corrective Action - Template

1. Defect ID
2. Defect Synopsis/Description
3. Defect Module
4. Defect Engineer [Defect Module Team]
5. Defect Disposition
6. Summary of Issue
7. Sequence of Events
8. Root Cause of Defect
9. Fix Details
10. Root Cause Category
11. Development Lead
12. Test Lead
13. Test Engineer
14. Next Steps
15. Corrective Actions
16. Root Cause & Corrective Action - Status

Fig. 7: RCCA Template

After every defect went through the analysis based on process and template, different root cause and corresponding actions were defined. Major root causes were: 1. Test doesn't exist [tester level] – at test engineering level tests are not designed in test plans which can catch such defects. 2. Test doesn't exist [developer level] – at development team level tests are not designed which can detect such defects at unit level testing. 3. Test doesn't exist [customer HW specific] – on internal available hardware such defects can not be detected. Only on customer specific hardware such defects are detected. No tests available for such specific hardware. 4. Test exists [execution missing] – tests are available in test database but was not executed at any stage of validation for software build release. 5. Test exists [test steps missing] – existing test having detailed, or modified steps would have detected the defect if executed during validation 6. Request for enhancement [RFE] – wishlist from customers in terms of new functionality or existing functionality modifications 7. Duplicate of tester defect – test engineers have already detected such defects in internal validation 8. Invalid [no actionable defect] – no actions can be taken on such defects 9. Customer specific customization – this is due to customer specific tools, scripts, applications and/or code implementation. Overall, root causes can be consolidated in six categories: Test doesn't exist, Test exist, Tester defects, Developer defects, Enhancement and Invalid. Based on these categorizations all defects data were root caused and were mapped based on defect origins. This can be understood in [Figure 8]:

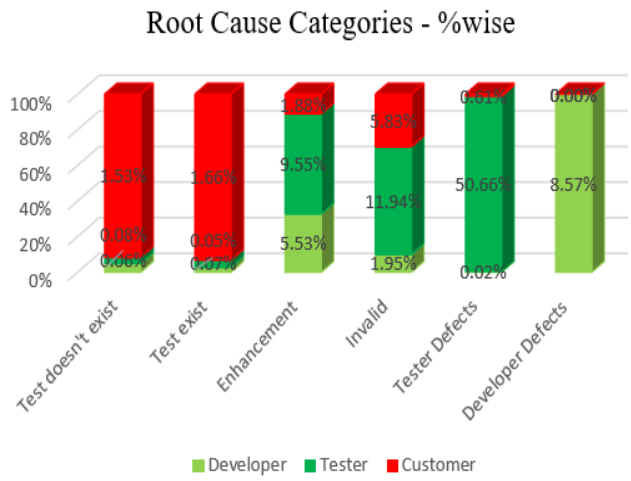

Fig. 8: Root Cause Categories Distribution

Against each root cause defined and categorized, based on RCCA template different corrective actions were also defined as part of analysis. These corrective actions are defined as: 1. Design new tests 2. Modify existing tests 3. Add existing/new tests into test coverage.

These tests when executed gives quality score. In traditional legacy software system model, legacy quality is analyzed and defined as Pass %, which can be mathematically represented as:

Pass % = [Total Tests Passed / Total Tests Executed] * 100

III. CONCLUSION AND FUTURE WORK

Based on root cause and corrective action analysis, it was observed that the most important attributes of defects which impact quality are: origin, severity, regression, priority and disposition. Root cause analysis helps define corrective actions which are further related to tests and test planning. Since legacy software system quality is based on tests, it can be safely concluded that, RCCA helps improve in:

1. Designing new tests – which will help detect defects.
2. Modifying existing tests – which will help detect defects.
3. Executing new and modified tests – which will help detect defects.

If all defects are detected in house by tester and developer itself, defects detected by customers will be lower, which will cause their experiences to improve and so the quality. Pictorially it be represented as in [Figure 9].

3.1. Future Study Focus

Future study will be focused on:

1. Analysis of defects based on 'Defect Module' and 'Software Stages', where defects are detected and fixed.
2. Categorization and defining all tests into 'Test Types' taken from test database of product under study. Map all tests to different 'Test Module' identified.
3. Create a relationship between defect and test – based on all attributes of defects and tests.

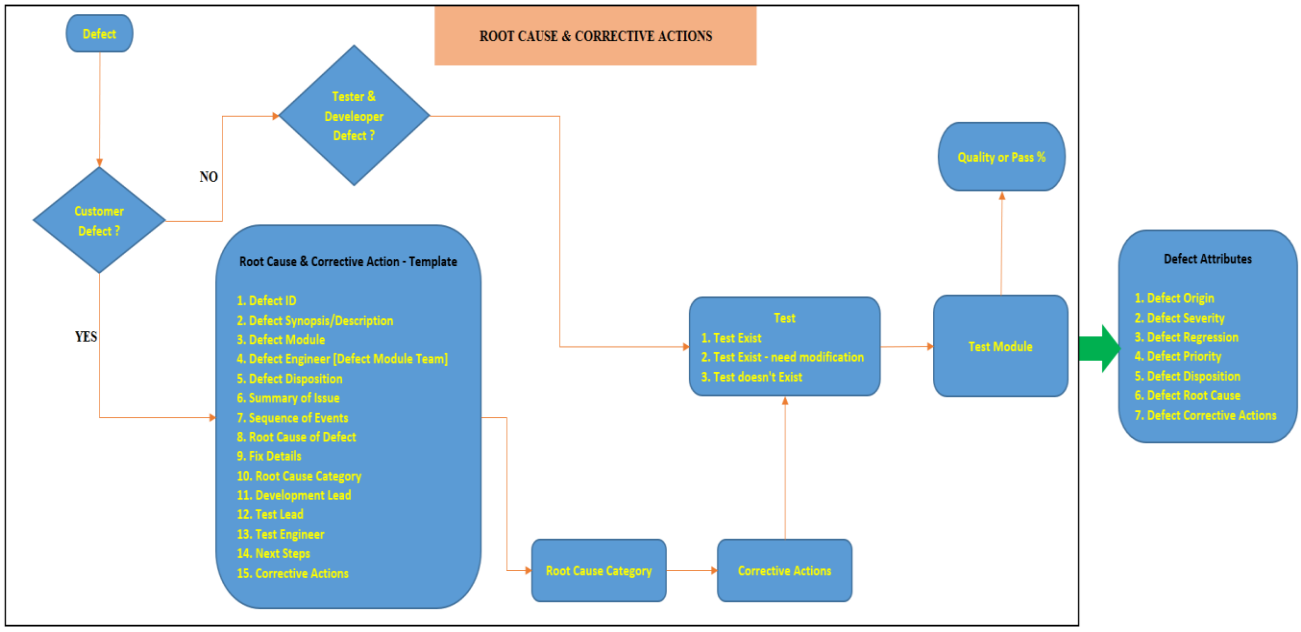

Fig. 9: Root Cause & Corrective Actions - Flow

4. Design a new software quality model based on these attributes and their inter-relationships.

IV. LIMITATION

In this study it is not analyzed based on which module and software stages defect is detected and fixed. In this paper tests of the product under study was not analyzed and how they are corelated with defects are not studied. All the points discussed in future study focus are limitations of this paper study. There are other attributes of software systems viz. complexity, code coverage, different other attributes of defects and tests, which are not taken into consideration for this research study.

ACKNOWLEDGEMENT

A sincere thanks to Prof. (Dr.) R. Kamatchi, Professor, Head, CSE, Amity School of Engineering and Technology, Mumbai, being a mentor and guide and for continuous help and support to come up this paper.

REFERENCES

1. Ian Somerville, “Software Engineering”, 8th ed., Addison-Wesley, pp. 5–6, 1982.
2. J.M. Juran, "Juran's Quality Control Handbook", McGraw-Hill, 1988.
3. Cagla Atagoren and Oumout Chouseinoglou, “A Case Study in Defect Measurement and Root Cause Analysis in a Turkish Software Organization”, In: Lee R. (eds) Software Engineering Research, Management and Applications. Studies in Computational Intelligence, vol 496. Springer, Heidelberg, pp. 55-72, 2014.
4. Abhishek Anurag, Dr. Kamatchi Iyer, “A Case Study of existing Quality Model based on Defects & Tests Management of Embedded Software System”, International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR) @ www.tjprc.org, ISSN (Print): 2249-6831; ISSN (Online): 2249-7943; Impact Factor (JCC): 8.9034; Index Copernicus Value (ICV): 60.28; NAAS Rating: 3.76; IBI Factor: 3.2. Paper Id.: IJCSEITRJUN20183, Vol – Issue: 8 – 2, pp. 15-30, Published: June 30, 2018; http://www.tjprc.org/view-achives.php?keyword=Abhishek+Anurag&jtype=2&from_date=&to_date=&journal=14.
5. Abhishek Anurag, “Study on Software Quality improvement based on Root Cause & Corrective Actions”, International Journal of Advanced in Management, Technology and Engineering Sciences @ http://ijamtes.org/, IJAMTES/319, Volume 8, Issue IV, A UGC Approved peer reviewed/referred Journal ISSN No: 2249-7455, Scientific Journal Impact Factor - 6.3, pp. 252-260, April/2018.
6. Marek Leszak, Dewayne E. Perry and Dieter Stoll, “A case study in root cause defect analysis”, Proceeding ICSE '00 Proceedings of the 22nd international conference on Software Engineering, ACM New York, ISBN:1-58113-206-9, pp. 428-437, June 04 - 11, 2000.