International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 6, June - 2014

Improving Software Quality using End user Interaction Oriented Defect Management Model

Ms Suman[1],
M.Tech (CSE), Department of Computer Science & Engineering,
Echelon Institute of Technology,
Faridabad, India

Dr. Manoj Wadhwa[2]
Professor & HOD, Department of Computer Science & Engineering,
Echelon Institute of Technology,
Faridabad, India

Abstract **- Every software product being tested before its release as no software can be built defect free. The main idea behind using Defect Management Process is to minimize the impact of defects in an organization and build a qualitative software product. Defect management process includes three steps: Defect detection, defect analysis and defect Preventation. At first step, the software product is tested until the defect identification process completed. In second step, Defect analysis working on previous detected defects with their root cause analysis. The defect analysis detects all those defects which are not find out earlier. The third step, Defect Preventation prevents the reoccurrence of defects in future. The main aim of this study is to establish a defect management process model which ultimately helps to reduce the defects and their impact and produce a good qualitative software product with the help of customer interaction.**

Keywords **Defect management process, defect analysis, defect detection and preventation, end user interaction.**

I. INTRODUCTION

A defect (or fault or bug) is a result of an entry of erroneous information into software [1]. The term defect refers to an error, fault or failure. The IEEE/Standard defines the following terms as Error: human actions that lead to incorrect result. Fault: incorrect decision taken while understanding the given information, to solve problems or in implementation of process. Failure: is inability of a function to meet the expected requirements [2] A defect [1] is defined as "a significant, unplanned event that occurs during testing of software process and it requires proper investigation and then correction. When a Defect is identified by a tester or user its complete information likes Defect id, status, resolution, and severity etc. recorded in a Defect Tracking system. This Defect report generated by the tester of software product, then Developer take a look about the Defect Report and tries to resolve the Defect. Software systems may have hundreds of defects. Defect tracking is the process of identifying defects in a product, (by inspection, testing, or recording feedback from customers),
Jones [1] lists the following as sources of defects: requirement defects, design defects, coding defects, documentation defects, test case defect and other defects.

II. SOFTWARE DEFECTS

Whenever a software product is examined or tested, different types of defects or bugs encountered in software [3]. These are:-

A. Requirement Defect
A mistake made in defining specification or requirements of customer needs for a software product. This includes defects in functional specifications, interface, design and test requirements and specified standards [3].

B. Design Defect
A mistake made in design of software product [11]. The defects found in functional descriptions, interfaces, logical defects in between code, data structures, error checking and standards.

C. Code Defect
A mistake made in implementation or coding of a program [3][13], the defects found in program logic, data definitions, handling interface between modules.

D. Document Defect
A mistake made in a software product publication [3] [14]. The defects found during document published.

E. Test Case Defect
A mistake made in test case causes the software product may produce an unexpected result [3].

F. Other Work Product Defect
Defects found in software product during development or maintenance of a software product [3] [14]. This includes test tools, compilers, configuration and other computer-aided software engineering tools.

III. DEFECT LIFE CYCLE

Defect life cycle shows a complete cycle in which a defect goes through various stages during its lifetime. It starts when defect is found and ends when a defect is closed [4].

The bug has different stages in the Life Cycle. The Life cycle of the bug can be shown diagrammatically as follows:

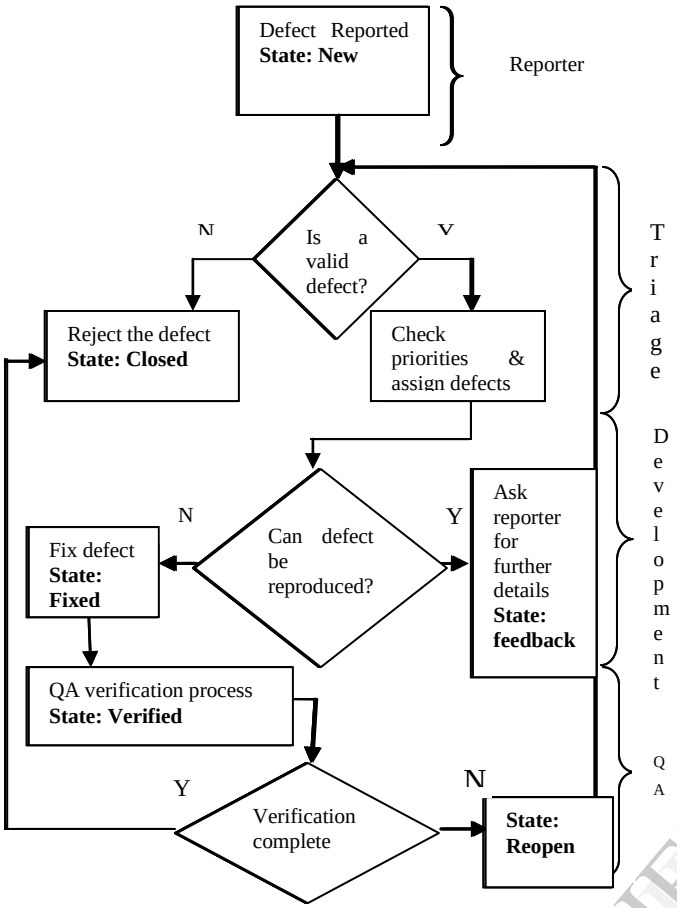

Fig. 1 Defect Life Cycle Workflow

- *New:* When a defect is reported by end user, tester or developer of the software. Its state is given as new.
- *Assign:* After the tester has posted defect the tester approves that the defect is valid then assign the defect to corresponding developer team.
- *Fixed:* When developer confirms that the information about defect is adequate then he/she assign defect status as 'Fixed' and the defect is passed to testing team.

- *Feedback:* If the defect is reproduced again then the developer ask reporter to send further details of the defect. After completion of feedback, the developer fixed the defect.
- *Verified:* The tester tests the defect again after it got fixed by the developer. If the defect is not present in the software, he approves that the bug is fixed and changes the status to "verified". Quality Assurance verification is performed in order to achieve Quality in work, activities being performed correctly and the software meets the requirements.
- *Closed:* Once the defect is fixed, it is tested by the tester. If the tester found that the defect no longer exists in the software, he changes the status of the bug to "closed". This state means that the bug is fixed, tested and approved.

- *Reopen:* If the defect still exists even after the verification complete then tester changes the status to "reopened". The bug goes through the life cycle once again.

IV.DEFECT MANAGEMENT PROCESS

The defect management process includes several steps. These steps implemented in an organization with standards and policies [3][13]. These steps are varied from organization to organization. The fig. shows general steps included in software management process:

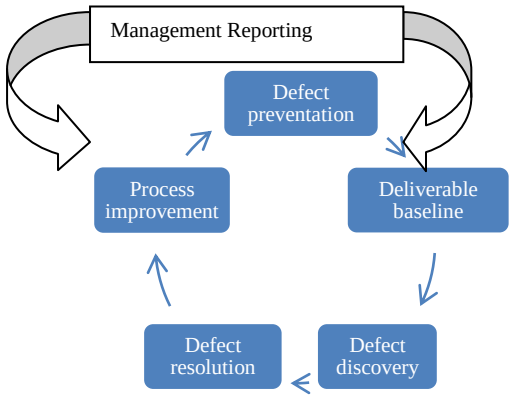

Fig.2 Defect Management Process

i) Defect Preventation

Defect Preventation process started with critical risks. The steps followed in this process are: identify critical risk, estimated expected impact and reduce the impact. The different techniques, methodology and standards get implemented for reduction of work.

ii) Deliverable Baseline

Milestones are established after which deliverables considered to be completed and ready for further development work [3]. A program should be considered as baseline when developer passed it for integration testing. Finally developers used requirement specification baseline for a technical design.

Errors in a deliverable are not considered defects until after deliverable are base lined.

iii) Defect Discovery

This step involves the identification of a defect. The defect discovery could perform by tester, developer or it may be end user. The step includes: find defect, report defect and acknowledge defect.

iv) Defect Resolution

Now, the developer fixes or resolves the defect and move the fix bug to next stage. This step include: prioritize risk, fix defect and report the defect resolution [3].

v) Process Improvement

In this step, the process a defect is identified and analyzed the way to improve the process to prevent the reoccurrences of defects in future [3].

V.DEFECT ANALYSIS

The main goal of defect analysis techniques is to analyze defects, identify their root causes and then developing the way to reduce these defects. The defects analyzed with the help of previous history of defect discovery. Some popular defect analysis techniques are: Orthogonal defect classification (ODC) [5] [6], Orthogonal defect classification computational code (ODC-CC) [5] [6], Modification Request (MR) Classification [8] and root cause analysis (RCA) [7], Fish bone analysis.

A. Issues related to Previous Work

Now different organizations are using defect management model presented by ITIL (Information Technology Infrastructure Library). Jantti marko and Miettien Aki in [5] describes different challenges to ITIL based processes. These are:

1) Defect recorded has many data fields which are hardly ever used.
2) It is not easy to combine the ITIL concepts with already existing defect management process in the organization.
3) No baseline available for known defects.
4) The number of known error idea is not detectable in the existing problem.
5) The testing support may lead to ambiguity as reported defects should have a link to test cases.
6) It is not easy to close many defects with one release because customized versions of product used by many customers.
7) In ITIL, it is difficult to define the right frequency of delivering defect fixes to different customers.
8) But ITIL modal does not describes a systematic way of testing and defect management activities [5].
9) The previous model does not conclude the customer interface in the entire management process.

B. Issues on Defect Classification

Various defect classification schemes for example ODC, MR classification and ODC-CC have been proposed previously but none of them came to be practical [12]. It is found in different studies that defect classification schemes are difficult to use as a general in practice [13]. They can be used in a specific environment or domain. Stefan Wagner in [12] proposes a set of challenges to defect classification schemes which are:

1) Interconnection of defects with different software artifacts in the existing classification schemes is available in different dimensions but none of them is clear.
2) There is no domain specific defect type distribution.
3) The work done on defect classification schemes partly related to software quality models.

ODC significant effects on the economics of root cause analysis by reducing the time and it also cover larger defect space particularly when the defect volume is large
[16]. ODC Improve software quality by using previously available data to decrease
Defects injected.

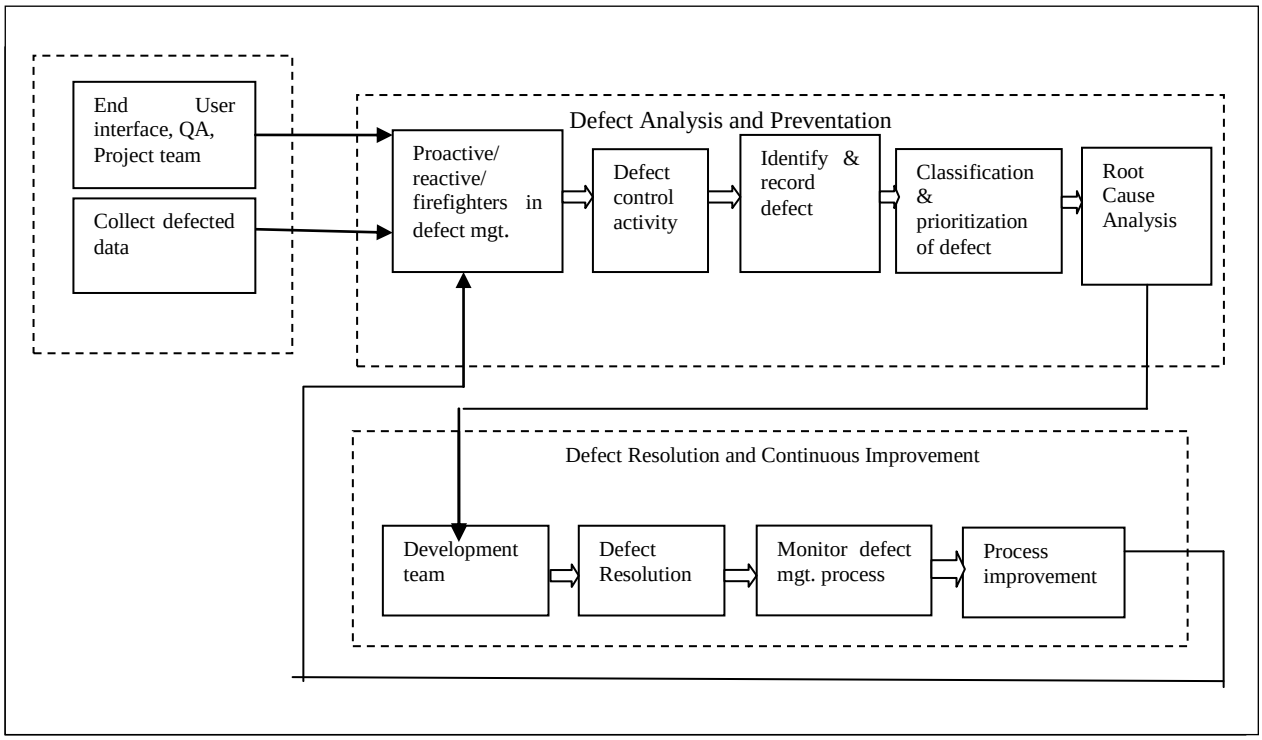

Fig. 3 Proposed Defect Management Model

C. The main limitations found in ODC are:

1) ODC cannot classify few defects such as GUI-type and data-type [10]
2) Normally ODC is useful in an organization which has a strong measurement system [4]
3) ODC require the capability to constantly group and analyze data over time; numbers of organizations are at lower maturity levels and they don't have this capability [10]
4) Updating of defect type makes it complicated to keep track of source defect data over a long period of time [10]

VI. PROPOSED MODEL

In Different studies different authors has explored the defect management process. Software Engineering Institute, Information
Technology Infrastructure Library and Quality Assurance Institute describe different types of defect management models [9] [10]. In this study authors have tried to propose a defect management process model. The propose model consist of three subparts which are a collection of defected data, defect analysis and prevention and last part is defect resolution and continuous improvement.

- *Customer interface, QA and Project Team*

The organization should collect the defected data from customer, QA and project team members in order to achieve a good defect fix rate in first time. The goal to collect the defect data from customer, QA and project teams is to resolve many defects as early as possible before delivering the product

- *Reactive/ Proactive/ Firefighters Defect management*

In Reactive approach, an Organization collected defect data to improve the way how the work should perform. Reactive approach is past oriented approach.
In Proactive approach, an organization analyses defect data continuously in order to prevent similar defects form occurrence in
the future [10]. The defected data shared
across the organization to reduce the reoccurrence of defects.
It includes the root causes of the defects and previous history of defects and process changes so that the defects won't occur again [10]. Steps followed in this approach are:

- Shifting defects from reactive responses to proactive responses.
- Perform root cause analysis to help decide what changes are required.
- Apply failure analysis and root-cause analysis to an effective and continuous process improvement.

Firefighters, the most basic approach which does not have any established processes for defect management [3].However firefighters do not use the defect data to affect any change in the software processes. They have defined processes for collection and handling of defect data.

- *Defect Control Activity*

Defect control activity starts when the analysis of defect data discloses repetitive occurrences or the analyzed defect does not match with the current problem.

- *Identify and Record Defect*

When defects occurred multiple times then identify all the defects and record them in the defect management system. The defect record linked with relevant problem. This will help to identify the defect solution or the strategy performed in the future. A defect has no meaning until they found defect reported and also the developer should acknowledge that the defect is found valid.

- *Classification and Prioritization of Defects*

Different classification schemes are used to classify the defects like ODC, ODC-CC, MR classification and RCA. Defects are classified by category, priority and their impact. The possible software defect categories for example can be functional, interface and algorithm etc.

- *Root Cause Analysis*

"Root-cause analysis is a group reasoning process applied to defect information to develop organizational understanding of the causes of a particular class of defects" [3]
There are many possible ways to analyze root-cause data.
Three approaches used in organization are:
□□One-shot root-cause analysis [11]
□□Post-project root-cause analysis [11]
□□Continuous process improvement cycle [11]

- *One -shot Root -Cause Analysis*

An efficient approach for organizations for categorization the defects that have not previously categorized done by one-shot root-cause analysis [3] [11]. This approach minimizes the amount of organizational effort invested on outsiders to perform the process.

- *Post -Project Root -Cause Analysis*

□The major difference between this process and the one-shot process is that organizations that start with the one-shot process have not previously collected causal data. Organizations that already collect failure-analysis data and have an understanding of their past defect patterns analyze their data and act on their results more efficiently [3] [11].

- *Development team*

Now, the development team has a number of solutions to current defects. The customer made a request to development team for implementing an efficient solution identified defect.

- *Defect Resolution*

Once the developer found that the injected defect is valid then the resolution process starts. While resolving the defect the developer should focus on fixing the defect. After resolution of defects developer must notify to all related parties about the defect status.

- *Monitor Defect Management Process*

Project management should continuously monitor the defect management process. Project management team should aware the working progress of the defect resolution process and impact of defects. The monitoring should be done based on actual requirements defined in the software requirement specification (SRS) document.

- *Process Improvement*

Most of the organizations ignored this step, due to this participants should go back to the phase from where the defect originated and brainstorm what caused the defect. After that they have to review the validation process in which the defect should be caught earlier. This step improves the review process.

VII. CASE$ STUDY

Table I represent the result of a project tested in an organization and chart 1 show the results into a graph form. The first row of table shows the four phases i.e. required to complete a project that is: Requirement analysis, Design, Coding or Implementation and Testing Phases. The user reported defect that are reported by end user of the product. Defect found in current phase estimated internally and user reported defects estimated externally or that are reported by end user. Defect density can be easily calculated once the numbers of known defects are found in a build. The defect density is calculated by using the following formula

Defect Density= Number of Known defects/KLOC

Where 'KLOC' stands for thousands of lines of code

Defect Removal Efficiency=Defects found in cur phase /User reported defects+ defects found in cur phase*100

Table 1. Four Builds of a project

Build	*User reported defects*	*Defect found*	*KLOC*	*Defect Density*	*DRE*
REQ.	120	350	58	6.0	74.5
DES.	90	315	70	4.5	77.7
IMP.	68	278	76	3.6	80.3
TST.	23	244	82	2.9	91.3

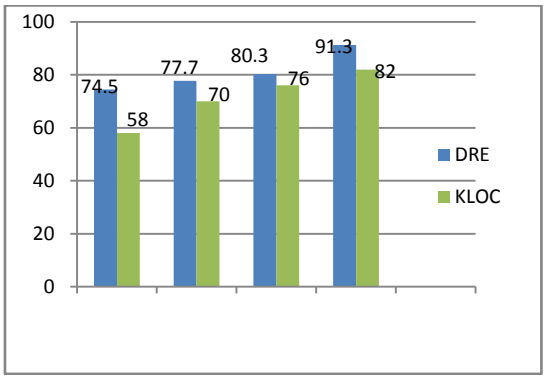

Fig. 4 Bar chart of Table 1

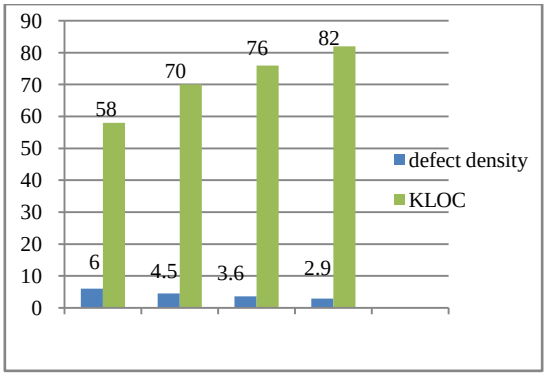

Fig. 5 Bar Chart of Table 1.

VIII.PROCEDURE

i) Identify and describe all the phases of a project-REQ, DES, IMP, TST with their defect found value in each phase.
ii) Define user reported defects in each phase and KLOC size.
iii) Calculate Defect Density as
DD=No. of defects found in cur phase /KLOC.
iv) Calculate DRE as
DRE=Defects found in cur phase /User reported defects+ defects found in cur phase*100
v) Software product is good when DRE has maximum value and DD has minimum value at the same point.

IX.CONCLUSION

An efficient defect management process is key element to build and produce a qualitative software system. To control the occurrence of defects in an organization, mainly three levels are used: Defect identification, Defect Analysis and Defect Preventation. Currently most of leading organization use ITIL model for defect management but this model don't support customer participation and criticize due to ambiguity in defect recording that affect its
performance .In this study, we have tried to propose a Defect Management Process Model in one of the case organization and found that proposed model is easy to understand and it supports the customer and development team interaction. The main aim behind this study is to establish a defect management process model in an organization to minimize the defects and produce a good qualitative software product.

REFERENCES

[1] Suma V, T.R. Gopalakrishnan Nair," Effectiveness of Defect Prevention in I.T. for Product Development", National Conference on Recent Research Trends in Information Technology, Bangalore, 2008
[2] P.K. Suri, Rajni Rana, "Defect Analysis and Preventation Techniques for improving Software quality", IJARCSSE, vol 3, issue 7, July 2013.
[3] Shruti Mittal, Kamna Solanki, Anuja Saroha," Better Management of defects for improving software process", IJCSMS, vol 11, issue 2, August 2011.
[4] Lisa Anderson, Brenda Francis, "The Bug Life Cycle", Journal of Defense Software Engineering, September 2003.
[5] A. A. Shenvi, "Defect prevention with orthogonal defect classification," ISEC'09, February 23-26, ACM, India, 2009.
[6] B. Robinson, P. Francis, and F. Ekdahl "A defect-driven process for software quality improvement," USA: New York, ACM, 2008.
[7] M. McDonald, R. Musson, R. Smith, D. Bean, D. Catlett, L. A. Kilty, and J. Williams, The practical guide to defect prevention, Washington: Redmond, Microsoft Press, ch. 11, 2008
[8] M. Leszak, D. E. Perry, and D. Stoll, "A case study in root cause defect Analysis," June, ACM, 2000.
[9] J. Marko and M. Aki, "Implementing a software problem management model, cases study", 2006, Springer Link.
[10] Hafiz Ansar khan "Establishing a Defect Management process model for Software Quality Improvement", IJFCC, vol 2, December 2013.
[11] Blanchard D., "Rework Awareness Seminar: RootCauseAnalysis,"March12, 1992.
[12] S. Wagner, "Defect classification and defect types revisited," July 20, 2008, Seattle, Washington, USA
[13] Feher P. and Gabor A. "The role of knowledge management supporters in software development companies" Software process improvement and practice, 11(3):251-260, June 2006
[14] Grady R.," Practical Software Metrics for project management and process improvement", Printce-Hall, Inc., 1992