Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Defect Analysis and Prevention Techniques for Improving Software Quality

Dr. P.K. Suri[1] , Rajni Rana[2]
[1]*Dean, Research and Development, Chairman, CSE/IT/MCA, HCTM Technical Campus, Kaithal, Haryana, India.*
[2]*HCTM Technical Campus, Kaithal, Haryana, India.*

Abstract: This research is a descriptive research on defect management process in software quality. Most large software products have elaborate quality control processes involving many tasks performed by different groups using a variety of techniques. The defects found are generally recorded in a database which is used for tracking and prioritizing defects. However, this defect data also provides a wealth of information which can be analyzed for improving the process. This paper, describe that when-who-how approaches for analyzing defect data to gain a better understanding of the quality control process and identify improvement opportunities. The proposed work is about to use defect tracking and defect prevention for quality improvement .

Keywords: Defect prevention, Defect tracking, Analysis, Features, Quality

I. INTRODUCTION

A defect is defined as: "An incorrect step, process, or data definition in a computer program." Software defects can be injected during any phase of the software development life cycle. Jones lists the following as sources of defects: requirement errors, design defects, coding defects, documentation defects and incorrect corrections. Lyu proposes four techniques which is used in the software development life cycle in order to reduce the amount of defects:

- Defect prevention takes aim to reduce the number of defects introduce while producing the software in the software development life cycle. This is done directly in any software engineering activity.
- Defect removal is to detect defects by software verification or software inspection. The main goal is to eliminate introduced defects. Strategies to achieve this may be dynamic analysis, or formal inspections of code.
- Defect tolerance is to provide continuous software service which satisfies given requirements despite a defect having occurred in the software.
- Defect forecasting is to estimate where new defects are likely to emerge in the software[1].

There are many software tools that play an important role in tracking defects of software and which are called "Defects Tracking Systems". Jalbert defined them as "Allow users to report, describe, track, classify and comment on bugs reports and feature requests". Defects Tracking Systems can be separated systems that can integrate, and contribute in software development process**.** They can keep, with details of defects reports and information associated with resolving it, in a database storage. Lethbridge, Singer and Forward indicated that developers view the defects tracking systems as important repositories of historical information . Furthermore, software defects data is an important source to the organizations for the software process improvement decisions and that "ignoring defected data, can lead to serious consequences for an organizational business". In addition "they may be part of an integrated suite of configuration management tools, where the status of the defect may act as a trigger or key for other events within the system". There is no doubt that software quality which is used in detecting defects, is one of the important factors for evaluating the software process development.The scope of this paper is to provide a comprehensive view on the principle of defect prevention, defect tracking system and defect analysis.

II. DEFECT PREVENTION

Defect prevention is an important activity in any software project. The purpose of Defect Prevention is to identify the cause of defects and prevent them from recurring. Defect Prevention involves analyzing defects that were encountered in the past and taking specific actions to prevent the occurrence of those types of defects in the future. Defect Prevention can be applied to one or more phases of the software lifecycle to improve software process quality [7].

A. Principles Of Defect Prevention

How does a defect prevention mechanism work?The answer is in a defect prevention cycle (Figure1). The integral part of the defect prevention process begins with requirement analysis – translating the customer requirements into product specifications without introducing additional errors. Software architecture is designed, code review and testing are done to find out the defects, followed by defect logging and documentation[8].

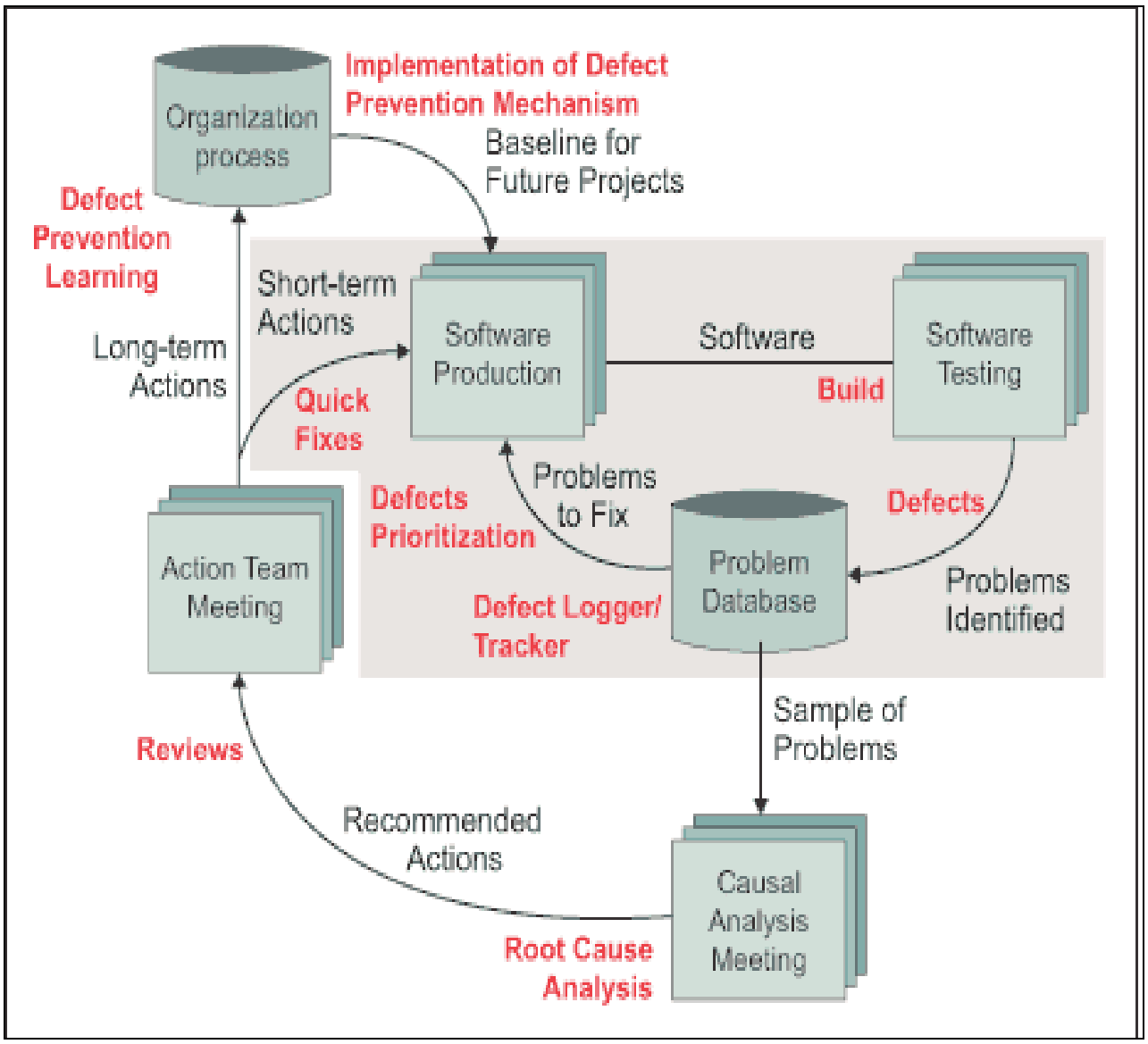

Fig1 Defect Prevention Cycle

The blocks and processes in the gray-colored block represent the handling of defects within the existing philosophy of most of the software industry – defect detection, tracking/documenting and analysis of defects for arriving at quick, short-term solutions. The processes that form the integral part of the defect prevention methodology are on the white background. The vital process of the defect prevention methodology is to analyze defects to get their root causes, to determine a quick solution and preventive action. These preventive measures, after consent and commitments from team members, are embedded into the organization as a baseline for future projects. The methodology is aimed at providing the organization a long-term solution and the maturity to learn from mistakes. Most of the activities of the defect prevention methodology require a facilitator[5]. The facilitator can be the software development project leader (wearing another hat of responsibility) or any member of the team. The designated defect prevention coordinator is actively involved in leading defect prevention efforts, facilitating meetings and communication among team and management, and consolidating the defect prevention measures/guidelines.

B. Defect Preventive Techniques And Practices

There are four types of defect prevention techniques which are described below in detail:-

1) Defect Prevention Through Error Removal: Defect through error sources can be removed in one or combination of following ways Train and educate the developers. About 40 to 50% of user programs contain non trivial defects[2]. Train the people and educate them in product and domain specific knowledge. Developers should improve the development process knowledge and expertise in software development methodology as well. Introduction of disciplined personal practices like clean room approach, personal software process and team software process reduces defect rate by up to 75%.

2) Defect Reduction Through Fault Detection And Removal: Large companies go for extensive mechanisms to remove as many faults as possible under project constraints. Inspection is direct fault detection and removal technique while testing is observation of failure and fault removal. Inspections can range from informal reviews to formal inspections.Testing phase can be subdivided as code phase of the product before the shipment and post release phase of the product. It includes all kinds of testing from unit testing to beta testing[10].

3) Defect Containment Through Failure Prevention: In this defect preventive approach, causal relationship between faults and resulting failures are broken and thereby preventing defects, but allowing faults to reside. Techniques like recovery blocks, n-version programming, safety assurance and failure containment are used. With the use of recovery blocks, failures are detected but the underlying faults are not removed, even though the off-line activities can be carried out to identify and remove the faults in case of repeated failures. N- version programming is most applicable when timely decisions or performance is critical such as in many real time control systems. Faults in different versions are independent, which implies that it is rare to have the same fault triggered by the same input and cause the same failure among different versions.For some safety critical system, the aim is to prevent accidents where an accident is a failure with severe consequence. In addition to above said quality assurance activities, specific techniques are used based on hazards or logical preconditions for accidents like hazard elimination, hazard reduction, hazard control, damage control.

4) Use Of Formal Methods Like Formal Specification And Formal Verification: Formal specification is concerned with producing consistent requirements specification, constrains and designs so that it reduces the chances of accidental fault injections. With formal verifications, correctness of software system is proved. Axiomatic correctness is one such method. Defect prevention based on tools, technologies, process and standards . Most of the company uses object oriented methodology which supports information hiding principle and reduces interface interactions, thus reducing interface or interaction problems. Likewise by following a managed process, ensuring of appropriate process selection and conformance, enforcement of selected product and development standard also prevents defect recurrence to a large extent[5]. Prevention of defects is possible by analyzing the root causes for the defects. Root cause analysis can take up two forms namely logical analysis and statistical analysis. Logical analysis is a human intensive analysis which requires expert knowledge of product, process, development and environment. It examines logical relation between faults (effects) and errors (causes). Statistical analysis is based on empirical studies of similar projects or locally written projects. Both the organization and the projects must take specific actions to prevent recurrence of defects. Some of the actions that are handled as described in Process Change Management Key Process Area are: - Goals, Commitment to perform, Ability to perform, Activities performed, Measurements and analysis and verifying implementations. The organization sets three goals like defect prevention activities which are planned, common causes of defects to seek out and to be identified, common causes of defects to be prioritized and systematically eliminated. The management owes certain commitment in order to get these goals into life. This commitment is seen as a written policy which is framed and implemented. The stipulated policy exists for the organization and for the project. It includes long term plans for funding, staffing and for the resources required for defect prevention. To improve the software processes and the products through DP activities, these results need to be reviewed and the actions are identified and addressed. For the DP to be able to perform, as per the Key Process Area, an organizational level team as well as the project level should exist. This may include teams from the Software Engineering.

III. DEFECT TRACKING SYSTEM

There are many software tools that play an important role in tracking defects of software and which are called “Defects Tracking Systems”.

A. Conceptual Framework For The Defects Tracking Model

Our model will focus, in details, on the phases of preventing defects and classifying them. The conceptual framework for classifying and tracking defects as shown in Figure (2), shows that the tracking system gives real historical information for maintaining the bugs through the development life cycle. Moreover, it concentrates on the bug examination, location and type factors for the insertion process of defect reports. Due to the highly exploratory nature of this study, all isolated

conceptual variables/factors only represent the initial ideas about the discussion of defects tracking phases and classification method to deal with in the future[5].

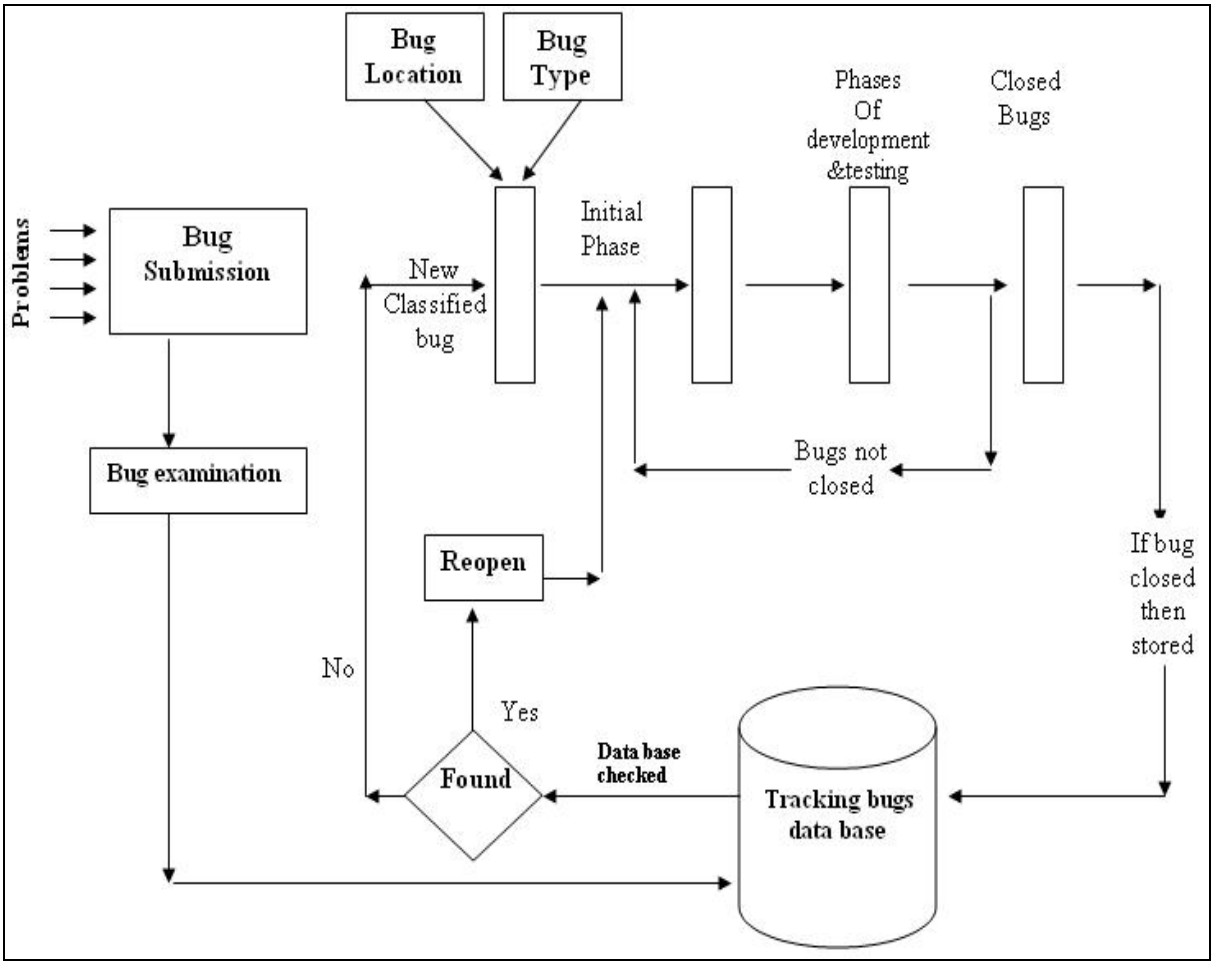

Fig2 Proposed Defect Tracking Model

IV. VARIOUS DEFECT TRACKING SYSTEM

A. Bugzilla

Bugzilla is a very popular, actively maintained and free bug tracking system, used and developed together with Mozilla, giving it considerable credibility. It is based on Perl and once it is set up, it seems to make its users pretty happy. It's not highly customizable. Bugzilla installations tend to look pretty much the same wherever they are found, which means many developers are already accustomed to its interface and will feel they are in familiar territory. Bugzilla has a very advanced reporting systems and you can create different types of charts including line graph, bar graph or pie chart. Bugzilla UI is strictly functional. There is nothing very nice about it, it provides plenty of functions within a small space, and in the beginning, the user can feel quite uncomfortable and lost; however, after discovering it, the user will find out that it is not very complicated and working with it is straightforward[5].

1) Advantages & Disadvantages:

- Bugzilla notifies users of any new or updated bugs by e-mail.
- Bugzilla supports basic time tracking.
- Bugzilla also supports a system of votes, in which users can vote for issues or features they wish to see implemented.
- Bugzilla is particularly complicated to install and maintain
- It supports large Projects.
- It doesn't have user-friendly interface.

B. Mantis

Mantis is a free web-based bug tracking system. It is written in the PHP scripting language and works with MySQL, MS SQL, and PostgreSQL databases and a web server. Mantis can be installed on Windows, Linux, Mac OS and OS/2. Almost any web browser should be able to function as a client. The main complaint is its interface which doesn't meet modern standards. On the other hand, it is easy to navigate, even for inexperienced users. There not exists some advanced features such as charts and reports. In short, the whole system is sloppily done, there are plenty of bugs and very little functionality.

C. Bugtracker.Net

BugTracker.NET is a free, open-source, web-based bug tracker or customer support issue tracker written using ASP.NET, C#, and Microsoft SQL Server Express. BugTracker.NET is easy to install and learn how to use. When you first install it, it is very simple to setup and you can start using it right away. Later, you can change its configuration to handle your needs. It has a very intuitive interface for generating lists of bugs[2]. It has two very useful features. First of them is a screen capture utility that enables you to capture the screen, add annotations and post it as bug in just a few clicks. The second feature is the fact that it can integrate with your Subversion repository so that you can associate file revision check-ins with bugs.

D. Bug-Track

Bug-Track is web-based defect and bug tracking software allows you to document, manage and assign all of your bugs and tasks and empowers you to organize your bugs, defects or issues into distinct projects. It can run on virtually any web-server like Microsoft, Linux, Unix, etc... Since it is an commercial application it is expected that it is better than other free products. But it isn't true. It has nothing new and better than other free bug tracking systems. One better thing is fact that it have more intuitive interface then others and that is his only benefit.

E. Redmine

Redmine is a flexible web-based project management web application. Written using Ruby on Rails framework, it is cross-platform and cross-database[10]. Redmine is open source and released under the terms of the GNU General Public License. Redmine is flexible issue tracking system. We can define our own statuses and issue types. It supports multiple projects and subprojects. Each user can have a different role on each project. Interface is very simple, intuitive and easy to navigate. Redmine is a very good recommended defect tracking system.

V. DEFECT ANALYSIS

Defect Analysis is using defects as data for continuous quality improvement. Defect analysis generally seeks to classify defects into categories and identify possible causes in order to direct process improvement efforts. Root Cause Analysis (RCA) has played useful roles in the analysis of software defects. The goal of RCA is to identify the root cause of defects and initiate actions so that the source of defects is eliminated. To do so, defects are analyzed, one at a time. The analysis is qualitative and only limited by the range of human investigative capabilities. The qualitative analysis provides feedback to the developers that eventually improve both the quality and the productivity of the software organization [8]. Defect Analysis is a process of classifying defects into categories and identifying possible causes in order to improve the processes. Defects are usually categorized as Blocker, Major, Minor or Enhancement; depending on the priority attached to fixing the bug Following table gives the details of the defects, which are all collected across the various projects for an analysis- *(http://software.isixsigma.com/library/content/c060719b.asp)*

Table I : Phase Wise Defects

Sl.No.	Phase	No. Of Defects	Identified Phase
1.	Requirement Analysis	698	Testing and Implementation
2.	Designing	3442	Testing and Implementation
3.	Coding	7724	Testing and Implementation
4.	Testing	78	Implementation
5.	Implementation	58	Implementation

From the above table, most of the defects were identified only on the Testing and Implementation phases i.e. at the later stages of the life cycle. So for correcting the above-mentioned defects , organization needs to spend lots of money and time on rework. It will increases rework effort significantly and capitulate to increase the Effort Variance and Schedule Variance [8]. Defects from Table I are analyzed using Pareto Analysis technique (Fig.3), it says that more than 80% of the defects are injected in coding and designing phases. So we can say that most of the defects are injected in coding and designing phases due to Incorrect Logic in design, Incorrect database design, incorrect report layouts, incorrect interface definition, incorrect logic, missing code, Exception handling error and navigation errors. That's why while designing the system, we need to consider aforementioned defect classification areas to reduce the rework time significantly.

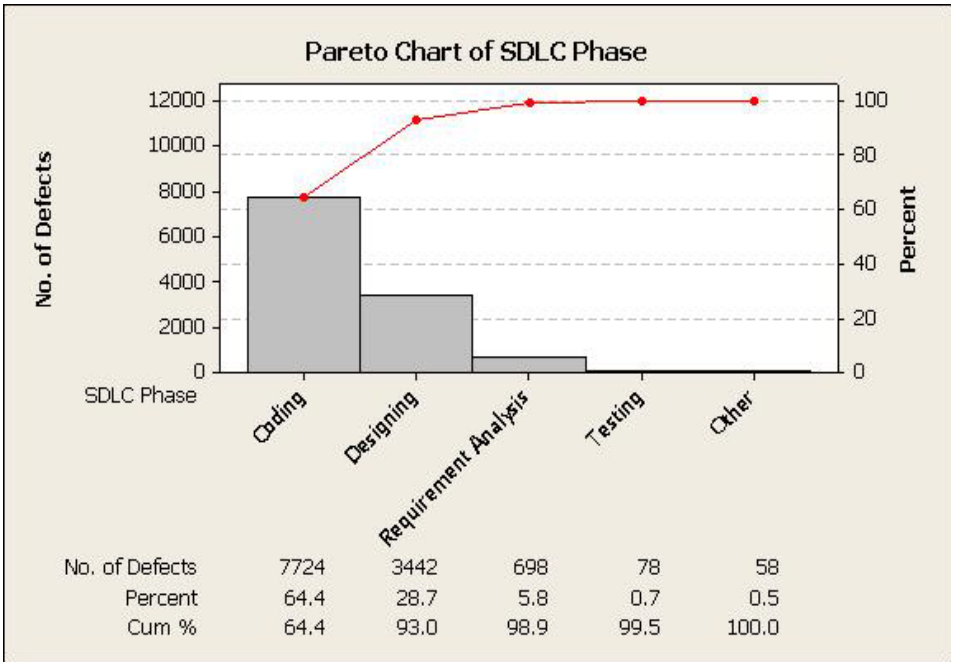

Fig.3 Defect Analysis Using Pareto Analysis

(*http://software.isixsigma.com/library/content/c060719b.asp)*

VI. CONCLUSION

This research focuses specifically on various defect tracking system with the respect to use of Defect Prevention for improving the software processes. As testing is always done after development of any software and it always show defect in it, which is resolved with the help of Defect Prevention by developer. Defect Tracking Systems still needs improvement in it and a lot of research is required to mature the Defect Tracking Systems.

ACKNOWLEDGEMENT

Sincere Thanks to HCTM Technical Campus Management Kaithal-136027, Haryana, India, for their constant encouragement.

REFERENCES

[1] Adeel, K., (Aug. 2005). Defect prevention techniques and its usage in requirement gathering – industry practices , Proceedings of Engineering Sciences and Technology, ISBN 978-0-7803-9442-1, SCONEST, IEEE Computer Society Publisher, pp 1-5.

[2] Ayman E.Khedr, Mostafa Sayed(2013),"Development And Evaluation Of A Defect Tracking Model For Classifying The Inserted Defect Data", European Scientific Journal April 2013 Edition Vol.9, No.12 Issn: 1857 – 7881 (Print) E - Issn 1857- 7431.

[3] Boehm, B. , Basili,V., (2001) Software Defect Reduction Top 10 List Published in: Journal Computer archive Volume 34 Issue 1, January 2001, IEEE Computer Society Press Los Alamitos, CA, USA

[4] Drago, J., (2011). "Role of Defect Management Software in Software Development Life cycle", Available at - http://ezinearticles.com/?Role-of-Defect-Management-Software-in-The-Software-Development-Life-Cycle&id=6205930.

[5] Fredericks, M., Basili ,V.,(1998). "Using defect tracking and analysis to improve software quality". Report DACS-SOAR-98-2, Experimental Software Engineering Group, University of Maryland, College Park, MD, USA.

[6] Megan Graham, 2005, Software Defect Prevention using Orthogonal Defect Prevention. http://twinspin.cs.umn.edufiles/ODC_TwinSPIN

[7] Suma V and T R Gopalakrishnan Nair , 2008, " Effective Defect Prevention Approach in Software Process for Achieving Better Quality Levels" Proceedings of World Academy of Science, Engineering and Technology, Volume 32 August 2008.

[8] Soni 2006, "Defect Prevention: Reducing Cost and Enhancing"Quality,isixsigma.compublishers,http://software.isixsigma.com/library/content/c060719b.asp

[9] Suma, V., Gopalakrishnan, T.R., (2010). "Impact Analysis of Inspection Process for Effective Defect Management in Software Development", American Society for Quality (ASQ) Journal, Software Quality Professional, USA, Vol. 12, No. 2.

[10] Florac, W., (1992). Software Quality Measurement : A Framework for Counting Problems and Defects, Technical Report CMU/SEI-92-TR-22.

