International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-3, Issue-3, May- 2015

Better Defect Analysis and Defect Prevention for Software Process Quality Improvement

D.Kavitha

Abstract— **Most large software products have good quality control process. Defect Prevention can be applied to one or more phases of the software development life cycle to improve software process quality. 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 future. Root cause analysis is the process of finding and eliminating the causes of defect, which prevents the problem from recurring. Pro-active defect prevention is used to control the defects with the experience of the previous projects. Reactive defect prevention identifies and conducts root cause analysis for defects meeting.**

Keywords— **defect, defect prevention, software quality**

I. INTRODUCTION

In Software development life cycle, defects are identified at various stages like Requirements Review, Design Review, Code Review, Unit Testing, Integration Testing, Functional Acceptance Testing, User Acceptance Testing and Customer Review.

P.K. Suri1 and Rajni Rana [5] defined defect as: “An incorrect step, process, or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner**”.**

“The defect management must be more proactive than reactive”.

Software defects data is an important source to the organizations for the software process improvement decisions and that ignoring defected data, results in great loss to the organization.

A. Defect

A Software **defect or bug** is a condition in a software product which does not meet a software requirement or customer requirement. A defect is an “error in coding or logic that causes a program to malfunction or to produce incorrect / unexpected results” [7]. Defect is defined as “A noted difference between the expected and actual behavior of application”. Bug is defined as “When a defect is reported and confirmed by development/ design team it is a Bug”. Issue is defined as “Any variation during the software development life cycle reported as issue”.

Manuscript received May 25, 2015.

D.Kavitha, Assistant Professor, Department of Computer Science and Applications, Nazareth College of Arts and Science, Chennai, India. d.kavithamoorthy@gmail.com

Wikipedia definition of Bug is: “A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design”. [8]

According to IEEE standard 729-1983 a failure is “an event in which a system or system component does not perform a required function within specified limits”. Failures are caused by faults.

A fault (or a bug) is “an accidental condition that causes a functional unit to fail to perform its required function”. Failures occur during the execution of a software program.

Defects may occur due to **design errors, logic errors** or **coding errors.** The defect may get introduced in any phase of the software development namely requirement analysis (RA), design (SD), coding (SC), testing (ST), implementation (SI) and maintenance of the system (SM). Debugging is the process of finding the causes of bugs. Bug tracking tools are used to track the bugs in the software project. Defect Report will contain detailed information’s about the defects.

II. SOFTWARE QUALITY & SOFTWARE QUALITY ASSURANCE

Software Quality

IEEE Standard 610.12-1990 [1] defines S**oftware Quality** as: “the degree to which a system, component, or process meets customer or user needs or expectations...”

Software Quality Assurance

IEEE Standard 610.12-1990 defines

- “A planned and systematic pattern of all actions necessary to provide adequate confidence that a software work product conforms to established technical requirements”.
- “A set of activities designed to evaluate the process by which software work products are developed and/or maintained”.

The traditional software quality assurance methods include testing, reviews, inspections, defect management models and techniques, such as defect prevention, the defect management process and root cause analysis method.

Quality Assurance consists of Auditing and Reporting functions of management. Software Quality Assurance involves the total software development process monitoring and improving the process, making sure that any agreed upon standards and procedures are followed and ensuring that problems are found and dealt with. It is oriented to defect prevention. Quality Assurance differs from Quality Control in that Quality Control is a set of activities designed to evaluate the quality of a developed or manufactured product.

As per Sakthi (2010), “A small amount of effort spent on quality assurance will save cost in terms of detecting and eliminating”. Software defects are time consuming and expensive.

The goal of quality assurance is to provide management with the data essential to be informed about the product quality, hence gaining insight and confidence that product quality is meeting its goals. If quality assurance identifies problems in the data provided, it is management’s responsibility to deal with the problems and apply the required resources to resolve the quality issues. Quality Assurance modifies development process to prevent the defects.

According to QAI – Quality Assurance Institute the defect management process consist of six elements.

- Deliverable base lining ,
- Defect discovery
- Defect prevention,
- Defect resolution
- Process improvement
- Management reporting

Defect discovery describes the techniques used to find defects. **Defect resolution** consists of prioritizing and scheduling the fix, fixing the defect and reporting the resolution. **Defect prevention** activity involves the analysis of defects that were encountered in the past and defining and implementing actions to prevent the occurrence of those defects in future projects. **Process Improvement** - Process improvement results in changing the existing quality manuals of Software Development life cycle process and results with the improved Software Development life cycle process and documents (Sakthi, 2010).

Software Quality Assurance Activities

The Software Quality Assurance group has responsibility for quality assurance planning record keeping, analysis and reporting. They assist the software engineering team in achieving a high quality end product.

The activities performed by an independent Software Quality Assurance group are,

- Formulating a quality management plan
- Applying software engineering techniques
- Conducting formal technical reviews
- Applying a multi-tiered testing strategy
- Enforcing process adherence
- Controlling change
- Measuring impact of change
- Performing SQA audits
- Keeping records and reporting

A. Statistical Process Control in Software Quality Management

The analytical tool from Total Quality Management TQM is the Statistical Process Control. This quality tool is useful for detecting the root causes of a defect and for classifying and prioritizing issues in a well established and ordered manner. Statistical Process Control is used to identify the process in control, and identify problem to make improvement.

B. Statistical Process Control Chart

Control Charts is a graph that monitors process quality. Control Charts process is within Statistical control limits. Control limits upper and lower bands of a Control Charts.

Process Variability

One of the successful project management is to achieve consistency and predictability, but there are always variations. These variations are due to unclear understanding of the requirements, changing technology and so many reasons. Variability refers to how the key metrics vary over time or over different projects. The Variability is within “reasonable limits” and an expected mean value. Lower Control Limit or LCL and Upper Control Limit or UCL are called as Reasonable limits. When the Variability exceeds these limits, they have to understand why this is happens and how to minimize such variation in future. Sample Statistical Process Control Chart is shown below.

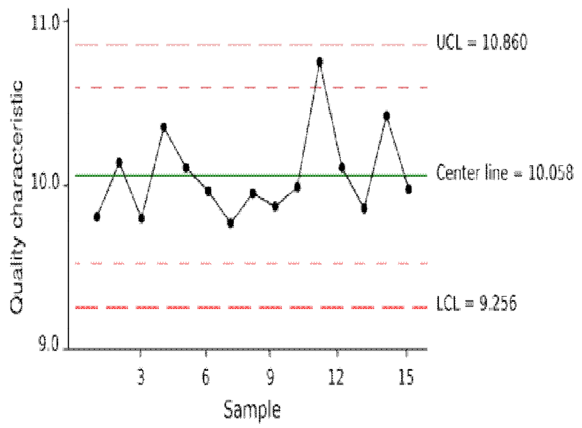

Fig 1: Statistical Process Control Chart

C. Software Process Improvement

IEEE definition for Process is: "A sequence of steps performed for a given purpose".

The definition for Software process is: "The set of Activities, Methods and Transformations that people use to Develop and Maintain Software and the Associated Products, for example: product plans, design documents, code, test cases and user manuals" **(SEI).**

Ian Sommerville [3] defines Software Process Improvement as "The process of making changes to a software [emphasis added] process with the aim of making the process more predictable or to improve the quality of its outputs". For example, if the aim is to reduce the number of defects in the delivered software, then they might improve the process by adding new validation activities.

In Software process improvement, there are five stages and they are Defect Identification, Defect Classification, Defect Analysis, Preventive Actions and Process Improvement.

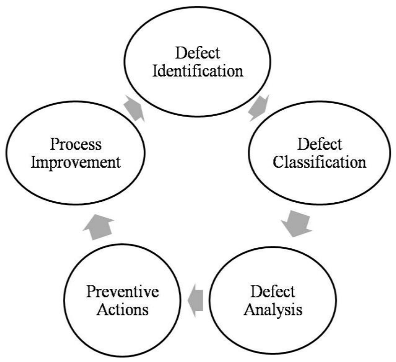

Fig 2: Process Improvement Workflow

III. DEFECT MANAGEMENT PROCESS

It is defined that, a defect management process as a well-defined and documented process for preventing, finding, and resolving defects. Defect management is a traditional way to see the process of handling defects. Defect management focuses on preventing defects and resolving existing defects.

Steps involved in Defect Management Process

1. Identification: This step involves the discovery of a defect.
2. Categorization: When a defect is reported, it is assigned to a designation team member to confirm that the defect is actually a defect as opposed to an enhancement next step is prioritization.
3. Prioritization: Prioritization is typically based on a combination of the severity of impact on the user, effort to fix, along with a comparison against other open defects. Prioritization is often handled by a formal change control board. Priority should be determined with representative from management the customer.
4. Assignment: Once a defect has been prioritized it is then assigned to a developer or other technician to fix.
5. Resolution: The developer fixes (resolves) the defect and follows the organizations process to move the fix to the environment where the defect was originally identified.
6. Verification: Depending on the environment where the defect was found and the fix was applied the **software testing team** or **customer** typically verifies that the fix actually resolved the defect.

A. Defect Identification (Defect Detection techniques)

Defects are found by preplanned activities specifically intended to find out the defects. Generally defects are identified at various stages of software life cycle through activities like Design review, Code Inspection, GUI review, function and unit testing.

Defect Detection techniques deal with how to find the faults:

a) Reviews and Inspection
b) Testing.

a) Reviews and Inspection

Software reviews identifies the issues earlier and cheaper than they could be identified by testing or by defect detection process.

Review is a form of static testing where people analyze the project or one of the project's work products rather than tools, such as a requirements specification. A review could be done entirely as a manual activity, but there is also tool support. Defects detected during reviews early in the life cycle are often much cheaper to remove than those detected by running tests on the executing code.

Types of the Reviews

- ➢ Code review
- ➢ Pair programming
- ➢ Inspection
- ➢ Walkthrough
- ➢ Technical review

Code review is systematic examination (often as peer review) of computer source code. The team examines a sample of code and fixes any defects in it. **Peer reviews** are composed of software walkthrough and software inspections. **Pair programming** is a type of code review where two persons develop code together at the same workstation. **Inspection** is a very formal type of peer review where the reviewers are following a well-defined process to find defects. Software inspection is a formal evaluation technique in which requirements, design and source code are examined to detect defects [6].

Walkthrough is a form of peer review where the author leads members of the development team and other interested parties through a software product and the participants ask questions and make comments about defects. **Technical review** is a form of peer review in which a team of qualified personnel examines the suitability of the software product for its intended use and identifies discrepancies from specifications and standards.

b) Testing

Software testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. Software testing is "a process of executing a program on a set of test cases and comparing the actual results with expected results". Testing cannot show the absence of defects, it can only show that software errors are present.

The Author Ehmer Khan (4), defines software testing as a process or a series of processes, design to make sure that computer code does what it was actually design to do and it doesn't do anything unintended. Software testing is a set of activities conducted with the intent of finding errors. It also ensures that the system is working according to the specification. White box testing is verification and validation technique which software engineers can use to examine their code works as expected.

Different types of testing happening during the software development are Unit Testing, Integration Testing, Functional Acceptance Testing, and User Acceptance Testing.

Unit Testing - Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

Integration Testing - Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design.

Functional Acceptance Testing - Functional testing refers to activities that verify a specific action or function of the code. Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test.

User Acceptance Testing - Consists of a process of verifying that a solution works for the user.

Once defects are identified then they are classified using first level of Orthogonal Defect Classification.

B. Defect Classification (Orthogonal Defect Classification)

After the defects are identified they are classified using first level of Orthogonal Defect classifications. An Orthogonal Defect classification (ODC) is the most prevailing technique for identifying defects where defects are grouped into types.

Defects Classification

Software Defects are normally classified as per:

- Severity / Impact (Defect Severity)
- Probability / Visibility (Defect Probability)
- Priority / Urgency (Defect Priority)
- Related Dimension of Quality (Dimensions of Quality)
- Related Module / Component
- Phase Detected
- Phase Injected

C. Defect Root Cause 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. The goal of RCA is to identify the root cause of defects and initiate actions so that the source of defects is eliminated.

The main goal of Root cause analysis is to identify the root cause of defects and initiate actions so that the sources of defects are eliminated. Root cause analysis is a group reasoning process applied to defect information to develop organizational understanding of the cause of a particular class of defects. The analysis should lead to implementing changes in processes that help prevent defects from re-occurring.

Members of the software engineering team can give the best suggestions for how to avoid such defects in future. Simple data analysis approach is to enter the data into an Excel sheet. Selected the defect report during the software maintenance and determined their error cause and root causes of the defects. There are many possible way to analyze the root cause of the defect data. The root cause facilitator is a person who runs the root cause analysis meeting. The software engineer's need to be skilled at meeting processes and dynamics and be familiar with software development and common defect types. Root cause facilitator analyzed the error cause of the defect data during the software maintenance.

D. Defect Prevention

Defect Prevention is an important activity in any software project. Defect prevention is also an essential part of software process quality improvement. Defect prevention is a process of identifying defects, their root causes, and also preventive measures have been taken to prevent them from recurring in the future, will lead to the better quality software product. Defect prevention is one such technique that is used in this software company and is required for a level 5 organization.

Defect Prevention can be applied to one or more phases of the software lifecycle to improve software process quality. Finding the causes and eliminating them are equally important. Once the root causes are documented, finding ways to eliminate them requires another round of brainstorming. The object is to determine what changes should be incorporated in the processes so that recurrence of the defects can be minimized. Monitoring the Defect Prevention Process is important, does the number of defects getting decreasing?

Fig: 3 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. Most of the activities of the defect prevention methodology require a facilitator.

The facilitator can be the software development project leader 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.

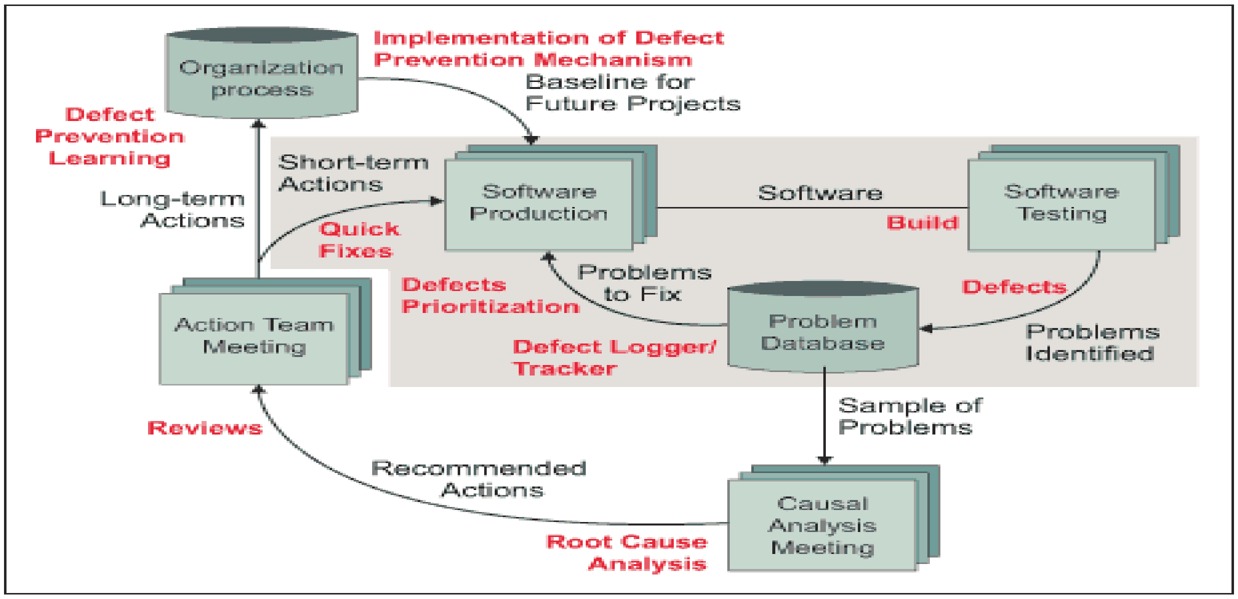

Fig 3: Defect Prevention Cycle [9]

Defect prevention tries to reduce the number of defects while producing the software in the software development life cycle. **Defect removal** is to detect defects by **software verification or software inspection**. The main goal is to eliminate introduced defects.

Defect prevention based on tools, technologies, process and standards. 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.

To improve the software processes and the products through Defect Prevention activities, these results need to be reviewed and the actions are identified and addressed. For the Defect Prevention 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.

IV. CONCLUSION

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. The process of the defect prevention methodology is used to analyze the defects to get their root cause to determine a quick solution and preventive action. Defect management reduces the cost of development of software product as previous reports get used to resolve the defects. Developing a good defect management process improves the quality of the software.

There are several difficulties involved in managing the defect but simultaneously it also has many benefits involved with it. The main being the organization which is implementing defect management, will have good reputation from customer. It is beneficial to integrate the defect management with software development process as it helps in removing the defects with almost every phase of development. A well planned defect management process is the main success factor for implementing software projects in time and in accordance with the budget. The key principle of the root cause analysis of a software defects is to reduce the defects to improve the software quality.

REFERENCES

[1] "IEEE Standard Glossary of Software Engineering Terminology" Software Engineering Standards Committee of the IEEE Computer society (1990)

[2] Sakthi Kumaresh, R.Baskaran,(2010), "Defect Analysis and Prevention for Software Process Quality Improvement", International Journal of Computer Applications (0975-8887) volume 8-No.7 October 2010.

[3] Ian Sommerville (2004), "Software Engineering", Addison-Wesley, 7th Edition.

[4] Mohd. Ehmer Khan, " Different Approaches to White Box Testing Technique for Finding Errors" International Journal of Software Engineering and Its Applications Vol. 5 No. 3, July, 2011.

[5] P.K. Suri1 , Rajni Rana, "Defect Analysis and Prevention Techniques for Improving Software Quality", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 7, July 2013.

[6].http://en.wikipedia.org/wiki/Software_quality#Measurement

[7]. http://softwaretestingfundamentals.com/defect/

[8]. http://en.wikipedia.org/wiki/Software_bug

[9]. [Mukesh] Mukesh soni, "Defect Prevention: Reducing costs and enhancing quality". February 26, 2010.

D.Kavitha is a Assistant Pprofessor in the Department of Computer Science and Applications, Nazareth College of Arts and Science, Chennai, India. She is M.C.A., M.Phil., (Ph.D).
She is also a Research Scholar ,(BHARATHIYAR UNIVERSITY), Area of interest is Software Engineering. Published papers in National and International conferences and journals. Eight years of experience in teaching undergraduate as well as postgraduate students.