Better Management of Defects for Improving Software Processes

Shruti Mittal[1], Kamna Solanki[2], Anuja Saroha[3]

[1]Student of M.tech, M.D University, UIET
Rohtak, Haryana, India
Shrutimittal25@gmail.com

[2]Assistant Professor in CSE Deptt., M.D University, UIET
Rohtak, Haryana, India
Kamna1604@yahoo.com

[3]Student of M.tech, M.D University, UIET
Rohtak, Haryana, India
Anuja.singh@yahoo.com

Abstract

Every software after development needs to get tested. No software can be built "Defect Free". After testing defects are reported by the use of a tool called "Defect Tracking System". Also Defects reported can be managed for enhancing the quality of software. This paper present the view of how defects are managed and the approach used for managing defect i.e. defect management process. Also it can be used for process improvement which means to prevent future occurrence of similar defects in processes.

Keywords: *Defect, Defect Management Process, Defect Analysis, Software Process Improvement with Defect Management.*

I. INTRODUCTION

Defect (or Fault or bug) is a result of an entry of erroneous information into software.[1] This could be due to an error in the requirements, design and architecture specifications. If these discrepancies are not identified during the review, then these may get translated into the introduction of an error into the application that needs to be identified during the testing phase. These can come out as defects with different severity (complexity) during testing.

A defect is a variance from specification. A defect is defined as "***any significant, unplanned event that occurs during testing that requires subsequent investigation and/or correction. Defects are raised when expected and actual test results differ***". [3]

When a Defect is identified by a tester or user, its related information (id, status and resolution, severity and priority and summary etc.) is recorded in a Defect tracking system. This information is called a Defect Report. Developer look at the Defect Report generated by tester and try 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), and evaluating these defects followed by prioritizing and managing them.

Using Defect tracking tool the following process is followed

- Logging in to the tool
- Defect Life Cycle
- Creating a defect
- Changing status of defects
- Generating metrics and reports

Raising a Defect: It is important that the tester verifies the defect by attempting to reproduce the failure and by seeking a second opinion and where possible obtain the initial acceptance of the Defect Manager[4].

The ***Defect Management Approach*** includes counting and managing defects. Defects are categorized on the basis of severity, and the number of defects in each category [6]. This count is used for planning the approach to be followed. Many software development organizations use tools to arrive at the defect leakage metrics (for counting the numbers of defects that pass through development phases prior to detection) and control charts to measure and improve development process capability. Also these defect data can be used for software process improvement (SPI)[8]. SPI is viewed as improving the software processes for the intent of increasing the quality of product [10].

II. SOFTWARE DEFECT

Whenever a software product is examined, different types of defects or bugs get encountered in software. These includes [2]:

- **REQUIREMENTS DEFECT:**

A mistake made in the definition or specification of the customer needs for a software product. This includes defects found in functional specifications; interface, design, and test requirements; and specified standards.

- **DESIGN DEFECT:**

A mistake made in the design of a software product [11]. This includes defects found in functional descriptions, interfaces, control logic, data structures, error checking, and standards.

- **CODE DEFECT:**

A mistake made in the implementation or coding of a program. This includes defects found in program logic, interface handling, data definitions, computation, and standards.

- **DOCUMENT DEFECT:**

A mistake made in a software Product publication [17]. This does not include mistakes made to requirements, design, or coding documents
.

- **TEST CASE DEFECT:**

A mistake in the test case causes the Software product to give an unexpected result.

- **OTHER WORK PRODUCT DEFECT:**

Defects found in software artifacts that are used to support the development or maintenance of a software product [17]. This includes test tools, compilers, configuration libraries, and other computer-aided software engineering tools.

III. DEFECT MANAGEMENT PROCESS

The defect management process include several steps When these steps get implemented in an organization, these have more detailed procedures with some specified standards and policies[11]. Steps in defect management process vary from organization to organization. Fig 1 shows general steps include in management process are:

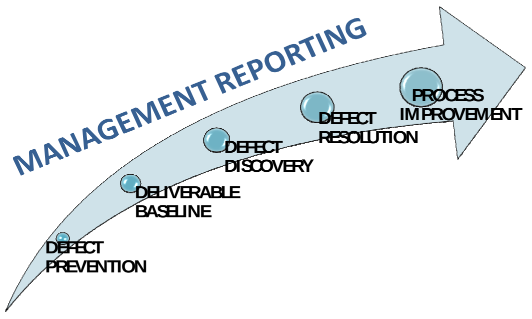

FIGURE 1

- **DEFECT PREVENTION:**

It is the process where different techniques, methodology & standard get implemented for reduction of risk.

- **DELIVERABLE BASELINE:**

Milestones are established after which deliverables considered to be completed and ready for further development work [7]. When deliverable is base lined, changes in it get controlled. Errors in a deliverable are not considered defects until after the deliverable is base lined.

- **DEFECT DISCOVERY:**

This step involves the identification of a defect. Hopefully, the person discovering the defect is someone on the testing team [13]. In the real world, it can be anyone including the other individuals on the project team, or on rare occasions even the end-customer.

- **DEFECT RESOLUTION:**

In this step, the developer fixes (resolves) the defect and follows the organization's process to move the fix to the environment where the defect was originally identified.

- **PROCESS IMPROVEMENT:**

In this step, the process in which a defect originated get identified and analyzed to identify ways to improve the process to prevent future occurrences of similar defects. Also the validation process that should have identified the defect earlier is analyzed to determine ways to strengthen that process.

The effectiveness of defect management system is influenced by the organizational culture it operates within[12]..If the organization consider the defects as the part of the process rather than taking it negatively seem to be able to deliver high quality software.

IV. SOFTWARE PROCESS IMPROVEMENT WITH DEFECT MANAGEMENT:

Software process improvement (SPI) is viewed as improving the software processes for the intent of increasing the quality of the software products [1,15] . This can be done through understanding the original software process and change it in order to increase the quality of the software products [2]. Grady claims software defect data is the most valuable source of information for software process improvement decisions . Further, the defect data provides a way of comparing improvements done against historic defect data in order to measure the effect of the improvements. He argues how ignoring defect data might yield dire consequences for business performance of an organization through reduced customer satisfaction and increased operational costs [18].

There are three ways organizations approach the handling of defects according to Basili and Fredericks.

Firefighters Approach
Reactive Approach
Proactive Approach

FIG 2

- **FIREFIGHTERS APPROACH :**

The most basic approach is the firefighters who have no established processes for defect management other than the ones required to keep track of them[7]. However, firefighters do not use the defect data to facilitate any change in the software processes. They have defined processes for collection and handling of defect data, but the defect data is never used.

- **REACTIVE APPROACH :**

The second strategy is to be reactive. Organization employing a reactive strategy uses the collected defect data to improve how they work.

- **PROACTIVE APPROACH :**

The third strategy is being proactive. An organization employing a proactive strategy analyses defect data continuously in order to prevent similar defects from occurring in the future [10]. They share defect data across the organization in order to elicit areas on where to improve.

One useful way to evaluate software defects is to transfer process learning from individuals to organizations. It includes brainstorming the root causes of the defects and incorporating what we learn into training and process changes so that the defects won't occur again [14]. There are five steps:

- Start shifting from reactive responses to defects toward proactive responses.
- Do failure analysis.
- Do root cause analysis to help decide what changes must be made.
- Apply what is learned to train people.
- Evolve failure analysis and root-cause analysis to An effective continuous process improvement Process

A. REACTIVE USE OF DEFECT DATA (A COMMON STARTING POINT):

After initial analysis, everyone reacts to defects either by fixing them or by ignoring them. This is often done with fast response to issues and by following up with patches or workarounds, when appropriate. There are some dangers that could occur if reactive processes aren't complemented with proactive steps to eliminate defect sources:

- People can get in the habit of emphasizing reactive thinking. This, in turn, suggests that management finds shipping defective products acceptable.
- Managers get in the habit of fixing defects late in development or after release.
- People place blame too easily in highly reactive environment.

B. FAILURE ANALYSIS (CHANGING YOUR MENTAL FRAME OF REFERENCE):

"*Failure analysis is the evaluation of defect patterns to learn process or product weaknesses*"[18].

The proactive use of defect data to eliminate the root causes of software defects starts with a change in mental frame of reference. The reactive frame generally focuses on single defects and asks "How much do they hurt?" It also considers how important it is to fix particular defects compared with others and asks "When must they be fixed?" The proactive frame asks, "What caused those defects in the first place? Which ones cause the greatest resource drain? How can we avoid them next time?

C. ROOT-CAUSE ANALYSIS PROCESSES:

"*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*" [2]

There are many possible ways to analyze root-cause data. Three approaches used in organization are:

- One-shot root-cause analysis[18]
- Post-project root-cause analysis[18]
- Continuous process improvement cycle[18]

- **ONE-SHOT ROOT-CAUSE ANALYSIS:**

A good starting approach for organizations that have not previously categorized their defect data by root causes is a one-shot root-cause analysis [16]. This approach minimizes the amount of organizational effort invested by using someone from outside the organization to facilitate 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 [4].

- **CONTINUOUS PROCESS IMPROVEMENT CYCLE:**

Some organizations have felt that root-cause analysis is so beneficial that they now use it to pursue continuous process improvement [9]. It appears to be a natural evolution from post-process root-cause analysis successes.

V. CONCLUSION

This paper gives a conceptual view of the defect management and processes used in it. It is very useful to manage the defects for improving the process of software development. Defect management reduce the cost of development of software product as previous reports get used to resolve the defects .There are several difficulties involved in managing the defect but simultaneously it also have many benefits involved with it. Use of Defect management improves the quality of software. The organization that implementing defect management will have a good reputation from customer. It is beneficial to integrate the defect management with software development process as it help in removing the defects with every phase of development.

REFRENCES

[1] Sommerville. I "Software Engineering 7th ed", Addison – Wesley publishing company, Boston, 2004
[2] Larman C, "Applying UML and Patterns", Prentice Hall PTR, October 30th 1997.
[3] Humpyrey WS, "A Displine for Software Engineering First (Ed):" Addison- Wesley Publishing company, Reading, 1995
[4] Shew hart WA, Deming WE, "Statistical Method from point of view of Quality Control ," publisher Dover ,New York , 1986.
[5] Singh, Rough, Gordon: "Standardization of the Quality Assurances 3rd (edition)", Saddle River, NJ: Prentice, Hall Inc, 1998
[6] Humphrey WS " Managing the Software Process", publishing company - Addison-Wesley, 1990
[7] Humphrey, W, "Introduction to Personal Process Improvement", Addison- Wesley, MA, 1996
[8] http://www.estylesoft.com/?id=317&pid=1
[9] Paul et al, "The Capability Maturity Model", publisher Addison-Wesley, Reading, 1995
[10] Pfleeger L, "Software Engineering Theory and Practice, (2ed)", Prentice Hall, Upper saddle River, 2001.
[11] "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
.[12] Johnson C, "The Benefit of PDCA", Quality Progress Vol 35, pp 120-121, 2002.
[13]Clements, Szy perski, "Component Software,"publishing Company Addison – Wesley, New York,200414] Cheeseman J, Dauds, "UML Components", publisher Addison, New York, 2004
[15]Daniel,Galin, "Software Quality Assurance", publishing company, Addison- Wesley, new York,2003
[16] Deutsch R, Williams R, "Software Quality Engineering : A total Technical and Management Approach", Printicel Hall international London, 1998.
[17] Grady R., "Practical Software Metrics for Project Management and Process Improvement", Prentice-Hall, Inc., 1992.
[18] Blanchard D., "Rework Awareness Seminar:Root-Cause Analysis,"March12,1992.